ASHMON -

A MONITORING PACKAGE FOR

PRIME 50 SERIES MACHINES

Paul Ashton
October 1984

CONTENTS

ABSTRACT

A major objective for all interactive systems is to provide adequate response time performance for terminal users. The best way to find the causes of inadequate response is to examine the behaviour of the components of response time plotted against workload. (Examples of components are : time running or waiting for the CPU, and time spent waiting for IO transfers). This report describes the development of ASHMON, a monitoring package designed to produce such plots for Prime 50 series computers.

In addition to software performance monitors, the package includes data interpretation programs which produce performance summaries and plots, a program which combines data from a number of monitoring sessions, and a program to configure the package on any Prime 50 series machine (apart from the P850). The report also includes discussion of experiments done to validate measurements obtained by the monitors.

ACKNOWLEDGEMENTS

There are many people who have made substantial contributions to the development of ASHMON. J P Penny conceived the underlying concepts used in the ASHMON monitors, and has given considerable guidance and support throughout ASHMON's development. Charles Brown has provided special rights to Canterbury's P750, which have been vital in ASHMON's development.

Barry Ananndale (MDP Ltd), and Chris Brice (Prime Computer), have aided ASHMON development by allowing ASHMON to be tested on different Prime configurations.

1 INTRODUCTION

Time-sharing computer systems are widely used. Such systems are sufficiently expensive to justify a lot of effort being spent on tuning them. A major tuning criterion might be that response should be adequate for terminal users during periods of peak terminal workload (typically during the mid-morning and afternoon). To improve terminal response, it is necessary to identify the major component(s) of response times for terminals when the system is under heavy load.

To determine the causes of poor response time, or to assess how response time might be reduced, one wants a monitor which can measure the "components" of response time, and plot them against workload. (Components are typically the times which processes spend running on, or waiting for, the various resources of the system). The objective for this project was to develop such a monitoring tool for the P750 at the University of Canterbury. The development has resulted in the ASHMON monitoring package, which includes monitors, data interpretation programs, and a package configuration program. This report describes the development of ASHMON, through the following stages :-

- basic design decisions
- determining factors in terminal response time
- investigating response ratio
- plotting of response time components against workload
- making ASHMON portable
- validating ASHMON

Although no formal performance evaluation studies have yet been undertaken with ASHMON, some useful and interesting results have been obtained. Examples of results obtained for the University of Canterbury P750 are given in Section 8. Interesting results have also been obtained for Mainland Data Processing's P550-II.

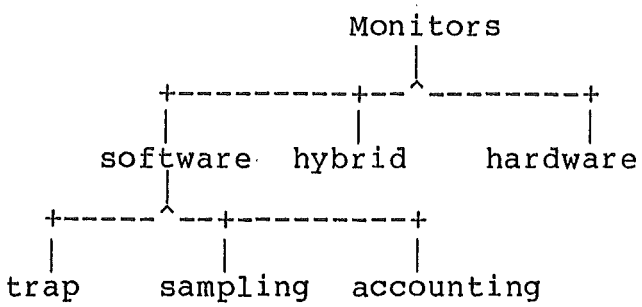
Because ASHMON monitors are heavily dependent on the architecture of the Prime 50 series machines, and the Primos operating system, a discussion of their features which are relevant in the development of ASHMON is given in Appendix 1.

This report concentrates on the design of ASHMON. The ASHMON users manual, a description of how to use the ASHMON package (written for Prime System Administrators and performance analysts), is included as Appendix 2 of the report.

2. BASIC MONITOR DESIGN ISSUES

2.1 Monitor types

MacGregor[1] summarised monitor types as follows :-



There are three basic types of software monitor. A trap (event) driven monitor is constructed by putting software probes within the operating system code, to record events of interest. Sampling monitors are programs, generally external to the operating system, which sample the state of the system at regular intervals. Accounting monitors derive their information from accounting data which is routinely kept by the operating system. In any software monitor the execution of the monitoring code produces some distortions in results.

A hardware monitor is traditionally independent of the system hardware, and is interfaced by connecting probes to the backplane of the processor. Information is derived by observing signals, such as voltage changes which show whether a unit is busy or idle. It seems likely that processors will be supplied with internal monitoring hardware in the future.

A hybrid monitor makes use of both hardware and software monitoring techniques.

2.2 Selection of monitor type

The goals of the monitor influence the monitor type selected. The design goal of ASHMON has been to produce plots of response time components against some workload measure, so that the major factors contributing to poor response time may be identified. The existing Prime monitoring tools (GEM and USAGE) cannot provide this sort of information. MacGregor[1] noted interest from Prime users in new performance evaluation tools for Prime computers.

To determine the components of response time, it is necessary to estimate the fractions of a given time interval which each terminal process spends in each response time state. Ferrari[2] has suggested that a software sampling monitor is well suited for this sort of analysis, and this monitoring technique is the one used in all ASHMON monitors.

The advantages and disadvantages of software sampling monitors are well documented in the literature[1,2,3,4]. There are many advantages in adopting this monitoring technique for ASHMON :-

- hardware monitors are not really usable because of the difficulty of attaching probes to modern processors.

- accounting data cannot provide the sort of information required for a response time breakdown.

- event detection monitors have an overhead when monitoring is not in progress.

- sampling monitors may be developed and used without operating system modification. Event driven monitors require insertion of software probes in the operating system code, which may greatly hamper monitor development[1].

- sampling monitors operate with less overhead than event detection monitors.

There are also problems associated with a software sampling approach. A problem common to all software monitors is that they require system resources - CPU time, IO time, and memory space. These requirements distort results obtained to some extent.

A second problem, specific to sampling monitors, is that sampling may not give an accurate reflection of system activity. There are two possible causes of this problem :-

(i) Sampling can become synchronised with system events. This introduces systematic error into results, which can be very hard to detect.

(ii) The regularity of sampling may be affected by workload fluctuations, which will in turn bias results. The sampling monitor must have high priority access to the CPU to ensure regular and unbiased sampling. Methods of giving ASHMON monitors high priority are described in Appendix 2.

Because sampling monitors may be implemented without altering the operating system, it is possible to port them to sites which do not have the source code for their operating system. (Most commercial sites do not have these source codes). A monitoring package with this aspect of portability is unusual, and desirable, as most software monitors are :-

(i) supplied as part of the system, or

(ii) developed for use on a specific site, or

(iii) supplied by independent performance evaluation companies, such as Morino, who look after installing and operating their monitoring packages. Most performance evaluation companies

concentrate on producing performance products for IBM equipment[3].

Therefore a major sub-goal in the design of ASHMON was to make it portable to any Prime 50 series machine. This meant that operating systems patches could not be used anywhere in the design of ASHMON.

2.3 Other design decisions

Two design decisions to be made when implementing a software sampling monitor are :-

(i) Whether to reduce data while monitoring and keep all data within the program, or whether to write results to secondary storage periodically.

(ii) Whether the sampling frequency should be fixed, or left as a parameter.

In early ASHMON monitors, raw data was written to secondary storage, with very little data reduction performed. This approach was found to incur too much CPU overhead and too much IO overhead. As a result, data reduction techniques have been implemented which keep allow data to be kept within the monitor until monitoring finishes. The CPU time required to perform IO was much greater than the extra CPU time required to perform the data reduction.

Decisions on the frequency of sampling, and the variability of this frequency, are important in the design of a sampling monitor. A high sampling frequency will result in a high monitoring overhead, but allows a realistic sample size (say 1000) in a short period. A high sampling frequency may also allow the detection of individual events. For example if disk IO was being monitored then, if the minimum IO time was 40ms and samples were taken every 30ms, every disk transfer could be detected by a sampling monitor.

A lower sampling frequency will lower the monitoring overhead, but will require longer periods in which to gather the same number of samples. Chances of event detection will be greatly reduced.

In ASHMON, there is no need to detect individual events. The decision on sampling frequency involves a trade off between overhead and the speed of gathering a reasonable number of samples. The sampling frequency decided on was 1 sample/second. This results in acceptable monitoring overhead (see section 7), and also gives a reasonably high sampling rate. Having a fixed sampling frequency may seem a little inflexible, but 1 sample/second is a good balance between overhead and sampling rate. A fixed sampling frequency also adds extra consistency when comparing results from different monitoring sessions.

Therefore the basic characteristics of all ASHMON monitors are that they :-

(i) are software sampling monitors, which
(ii) reduce data as it is gathered, and
(iii) sample at a frequency of 1 sample/second.

3 RESPONSE TIME COMPONENTS

The next problem was to find a method of determining the "state" of a process (i.e. the current activity that a process is engaged in), in order to be able to determine the length of time that a process spends in each of its possible states. A monitor may then be constructed to produce reports on response time components. Obviously, response time states (those states which the user process goes through during response time) are of particular interest.

While the idea used in this monitor may be applicable to most time-sharing systems, identification of process' states must be done in ways which are very specific to the machine in question. In the following discussion the reader is assumed to have knowledge of Prime architecture and software, as summarised in Appendix 1.

3.1 Determining the state of a user process

ASHMON monitors record data for user processes only (see 3.2.3). Therefore, the only states of interest are states which user processes can take. J P Penny suggested that a breakdown could be achieved by monitoring the semaphore words in each process' PCB. These two words contain either :-

- the virtual address of the semaphore that the process is waiting on, or

- an indication that the process is on the ready-list.

Therefore, by examining these two words, the state of a process may be determined. The first task was to find those semaphores on which user processes commonly wait. Many semaphores in Primos are seldom used and, for monitoring purposes, can be ignored. The following groups of semaphores were found to be significant :-

(1) Ready-list, HIPRIQ, ELIGQ and LOPRIQ

Processes queued on the ready-list, or waiting on the HIPRIQ, ELIGQ or LOPRIQ semaphores, are running on, or queueing for, the CPU. These states will make up a significant part of response time.

(2) Semaphores in disk QCBs

Processes waiting on semaphores within disk QCBs are waiting for disk transfers to be performed. This state will also be common.

(3) DSKBLK semaphore

Processes waiting on this semaphore are waiting for a disk QCB to become free. This state is relatively rare as there are 17 disk QCBs available in Primos of Rev. 19, which means that 17 disk requests must be pending before DSKBLK will be waited on.

(4) MTlSEM semaphore

Processes on this semaphore are waiting for the completion of magnetic tape operations. This state occurs sporadically, as tape usage (at Canterbury anyway) tends to be sporadic.

(5) Nl locks

Primos contains a group of locks to protect system data areas which allow N readers or 1 writer at any one time. The locks protect such data as UFD, file system, and paging data. This state tends to make up a noticeable component of response time on most systems.

(6) NTWAIT semaphores

Each user process has an associated NTWAIT semaphore, which it waits on whenever it is waiting for response from a network. The frequency of this state depends on the usage of any networks that the particular Prime is connected to.

(7) ASRSEM semaphores

Each user process also has an ASRSEM semaphore, which it waits on when it is waiting for input from a terminal, or from a command file. Waiting for input is the most common state for a terminal process to be found in.

(8) CLKRNG semaphores

CLKRNG is a group of 10 semaphores used to implement process suspensions for specified time periods. User processes are often found suspended on CLKRNG.

(9) USRSEM and NBRSEM semaphores

The user (USRSEM) and numbered (NBRSEM) semaphores are available to users, and to the operating system for general use. Thirty-three of the numbered semaphores are reserved for operating system use, and the remainder are available to all users.

(10) PHMSEM

PHMSEM is the semaphore phantoms wait on when they are not logged-in. This state is very common for phantom processes.

3.2 Extra breakdowns required

A user process can therefore be classified into the following states by the value of the semaphore words in its PCB :-

- on the ready-list,
- waiting on a processor semaphore,
- waiting for a disk transfer,
- waiting for a free disk QCB,
- waiting for response from a tape drive,
- waiting on an N1 lock,
- waiting for network response,
- awaiting terminal or command file input,
- suspended for a time interval,
- waiting on a user or a numbered semaphore,
- a logged-out phantom waiting to be logged in.

This breakdown has certain deficiencies. These are :-

(i) It is not possible to divide processes on the ready-list into :-

(a) the process "running" on the CPU (i.e. the process that would have been running if the monitor had not been), and

(b) the remaining, queued, processes.

(ii) There is insufficient disk IO information available. The only information available from the semaphore words is that a process is waiting for the completion of a disk transfer. This level of detail is clearly insufficient as, at the very least, a distinction between paging and file IO transfers is required.

(iii) It is not possible to distinguish between terminals that are logged-in and waiting on their ASRSEM, and those logged-out and waiting on their ASRSEM.

The resolution of these problems is described below.

3.2.1 Determination of the process "running" on the CPU

Determination of the process which would have been running on the CPU if the monitoring process had been absent is fairly easy. ASHMON monitors run on user priority 3, or the SP1 process' priority (see Appendix A, and 5.5). Whenever an ASHMON monitor is collecting data, it is the running process. The next process to be run is determined by scanning the ready-list from user priority 3 down (backstop will be found if no other user processes are on the ready-list). If the monitor is running on user priority 3, it checks the link word in its PCB to see if there are any other processes at user priority 3, and then scans the ready-list from user priority 2.

Processes waiting on the processor semaphores (HIPRIQ, ELIGQ, LOPRIQ) are not considered to be runnable.

The P850 has a dual stream processor, which would make determining the running process (or processes) somewhat different from all other 50 series machines, which have single stream processors. ASHMON could be adapted to run on an 850, but an 850 has not yet been available.

3.2.2 Further break-down of disk information

Detailed disk IO information is available in the disk QCBs, the disk queue headers, and the device busy bits. The semaphore words in a process' PCB allow identification of the QCB allocated to that process, when the process is waiting for a disk request. The useful pieces of information in the disk QCBs are the physical device number for the transfer, and the QCB link. The physical device number specifies the logical area that the transfer is for. As paging areas are logical areas, the physical device number gives adequate information on the destination of the request.

From the above information, it is also possible to decide which requests are presently being acted upon, and which are still queued. The requests being acted upon (i.e. those running) can be determined for each sample in the following manner :-

The device busy bits for each controller are examined. Whenever a busy drive is indicated, the QCB list for the controller (pointed to by the relevant disk queue header) is scanned until a QCB is found which contains a request for the busy drive. The process associated with this QCB is deemed to be running a disk transfer.

Therefore, processes waiting for disk IO can be classified as to which physical device they are referencing, and whether the request is being acted on or is queued.

In a few cases, interrupts by disk controller processes can result in samples with inconsistent disk IO data. For example if a drive's device busy bit is set, but no request is found for that drive. Samples are also discarded if a page fault occurs while the monitor is gathering the sample's data. In both cases samples are discarded as they are not an accurate snapshot of the state of the system. All ASHMON monitors keep counts for both kinds of sample rejects.

3.2.3 Distinguishing logged-out terminals

A further piece of information stored in the PCB allows distinction between logged-in and logged-out terminals, where processes are waiting for terminal input. The program counter (PB) in a logged-out terminal's PCB, is the address of the WAIT on the ASRSEM in the logout routine. By comparing this address with the terminal process' PB, it is possible to determine whether a terminal waiting on its ASRSEM is logged-in or not.

3.3 Construction of a monitor to break-down response time

With the above information a sampling monitor can be constructed to give a break-down of response time. It needs to access :-

- user PCBs
  - semaphore words
  - PB words
  - link word in monitor process' PCB
- disk QCBs
  - link word
  - physical device number word
- ready-list
- device busy bits
- disk queue header pointers

A detailed response time break-down is kept for terminal processes only, because in general terminal response is of overriding concern on an interactive system such as the Prime. No information is kept at all for the interrupt processes, or the console process, as they run at priorities above user priority 3, and cannot, therefore, be accurately observed by a monitor whose process is running at user priority 3. Although response times for phantoms are not of great significance, a record of their effect on workload is of interest, and information on this is kept.

A "response vector" is used to keep counts of the frequencies for the various user states. The vector maintains counts for the following states of terminal processes :-

- waiting for terminal input from logged-in terminals
- processes suspended on CLKRNG
- "running" on the CPU
- queued on the ready-list
- waiting on HIPRIQ
- waiting on ELIGQ
- waiting on LOPRIQ
- running disk requests for each physical device
- queued disk requests for each physical device
- waiting for free disk QCBs
- waiting on the magnetic tape subsystem
- waiting on N1 locks
- waiting on numbered and user semaphores
- waiting on any semaphore not covered above

Two counts of phantom activity are maintained :-

- logged-in phantoms
- "active" phantoms (Meaning of "active" given in 3.3.1)

In every sample the following activities take place :- the relevant data is collected, a response vector for the sample is derived from the data, and, unless the sample is rejected, the sample response vector is added into the monitor's total response vector. In a sample, a terminal is deemed to be running on the CPU if a terminal process is the next runnable process. The monitoring process ignores its own PCB as it will always find itself on the ready-list.

The result of a monitoring session is a response vector representing the accumulated sample response vectors for the period sampled, along with various counts of sample rejects.

3.3.1 Deriving a Performance Summary from a Response Vector

As mentioned previously, the states of most interest are the response time states. A division of terminal time into categories of response time (active) states, and other (idle) states is therefore desirable. Most of the terminal states listed above are response time states, apart from the following three :-

(i) waiting on the terminal input semaphore. This time is obviously the system waiting on the user, and therefore this is not a response time state.

(ii) suspended on CLKRNG. Investigations showed that terminal processes found waiting on CLKRNG were generally suspended in an attempt to let the output buffer of the terminal empty. This state is the system waiting for the terminal, and so it is not really a response time state.

(iii) waiting on user and numbered semaphores. These semaphores are hard to classify as their uses are determined by whoever is using them. They are put into the category of non-response time states as a miscellaneous item.

Terminal processes found in response time states are classified as active processes. A similar classification is applied to phantom processes. A logged-in phantom is classified as active unless it is :-

(i) waiting for input

(ii) waiting on a numbered or user semaphore (batch phantoms, and printer phantoms have an operating system numbered semaphore each, which they wait on when they are non-active)

(iii) suspended on CLKRNG (the card reader phantom waits on CLKRNG when idle)

(iv) waiting on network response. This classification of network waits differs from the classification for terminals. The reason is that file-transfer phantoms wait on network semaphores when idle, and these phantoms are hard to distinguish from any others. As phantom processes are only categorised as logged-in or active this classification problem is unimportant.

With these classifications, and the data given by a response vector a performance summary may be prepared. The summary was the first information provided by ASHMON monitors, and it gives details of response time components, total (terminal) time components, a detailed analysis of terminal disk traffic, various user measures, and an estimate of response ratio for the period monitored (response ratio is discussed in Section 4).

An example of a performance summary, with explanations is given in the ASHMON users manual which is Appendix 2 of this report.

4 RESPONSE RATIO

Mean response time is an index commonly used to describe response. Mean response time is, however, a workload-dependent index. For example a mean response time of 10s for major computations is good, and a mean response time of 10s for a list directory operation is bad, but in both situations response time is the same. On a system with a diversity of uses, mean response time may not be a particularly meaningful index.

Indices which are more workload-independent have therefore been sought. Response ratio, an index which estimates the ratio of actual response time to "optimal" response time, has proved popular. Optimal response time is the time an interaction would take if it had no competition for system resources from other interactions. Response ratio is a factor showing the increase in response time due to multi-programming[10]. Abrams and Treu[5] use the above definition for what they call the "interference ratio of response time".

A related index is stretch factor (stretching factor of response time), defined by Ferrari[2] to be the actual response time divided by CPU time required. Barney et al[9] describe stretch factor as a useful measure of delay in receiving service.

4.1 Calculating Response Ratio

Consider response ratio RR, defined by

$$RR = \frac{\text{actual response time}}{\text{optimal response time}}$$

To calculate response ratio from a response vector, the states which occur in an optimal response (the optimal response time states) must be identified[10]. The optimal response states do not include queueing states (no queueing is required during an optimal response), or paging IO states (no paging occurs during an optimal response).

It is not so clear, however, whether waiting for network response should be included as an optimal response time state. Ideally, an estimate for optimal response time for an interaction should include the minimum reply times for any requests it makes of any network. At present optimal times for network requests are impossible to measure, and so waiting for network response was not included as a component of optimal response time.

Optimal response time is estimated from the time spent in the following response time states :-

- running on the CPU,
- running disk requests on file (non-paging) disk partitions.

Response ratio is therefore calculated by the formula

$$\text{response ratio} = \frac{f(RT)}{f(CPU) + f(IO)}$$

where f(RT) is total time spent in response time states,
f(CPU) is time spent running on the CPU, and
f(IO) is total time spent running file IO.

4.2 RR monitor

Since response ratio could be estimated the from data gathered, an interactive monitor called RR was constructed to give regular estimates of response ratio. The monitor requires two results to calculate response ratio :-

- the frequency count of response time states, and
- the frequency count of optimal response time states.

A count of active users (active terminals + active phantoms) is also kept, to give an idea of the workload present for the period monitored. The RR monitor is small as it needs only to distinguish between processes which are active, in response time state, or in optimal response time state. The RR monitor is the first of the current ASHMON monitors to be discussed in this report. Its use and output format are described in Appendix 2.

4.3 Accuracy of estimate for response ratio

This method of estimating response ratio is subject to a number of possible errors. The major causes are that :-

(i) Software monitors are never totally accurate. Sampling monitors are also subject to statistical uncertainties, particularly with small sample sizes. Both of these factors contribute to uncertainties in the accuracy of figures collected in the response vector, and the counts accumulated by RR.

(ii) In the Prime architecture, the CPU and IO subsystem share a single path to main memory. When DMA IO is in progress the CPU faces contention for main memory cycles. Loffler[8] states that when DMA is proceeding at its maximum rate (8 Mb/s) the power of the Prime 750 processor is effectively halved. Observed deteriorations do not approach this, as DMA IO speed never reaches 8 Mb/s for any length of time, but on the Canterbury P750 CPU time debitted to an interaction increases by 10% or more when the system is under load. This means that the CPU run component of optimal response time will be slightly overstated.

(iii) As workload increases, demands for file buffers increase. Therefore, at higher workloads more file IO transfers will occur than would have been needed under optimal conditions. The file IO run component of optimal response time may therefore be overstated.

(iv) As discussed earlier, the determination of optimal network request times is difficult, and is not attempted by ASHMON monitors. If waiting on network requests forms a significant part of response time their non-appearance in the calculation of optimal response will cause optimal response time to be understated.

The above sources of error should not unduly affect the usefulness of the response ratio estimate, as experience suggests that figures for response ratio have been indicative of performance. Response ratios (and response indices in general) are usually used for comparisons, so as long as response ratio is measured in a consistent manner, the absolute accuracy of measurement is not vital.

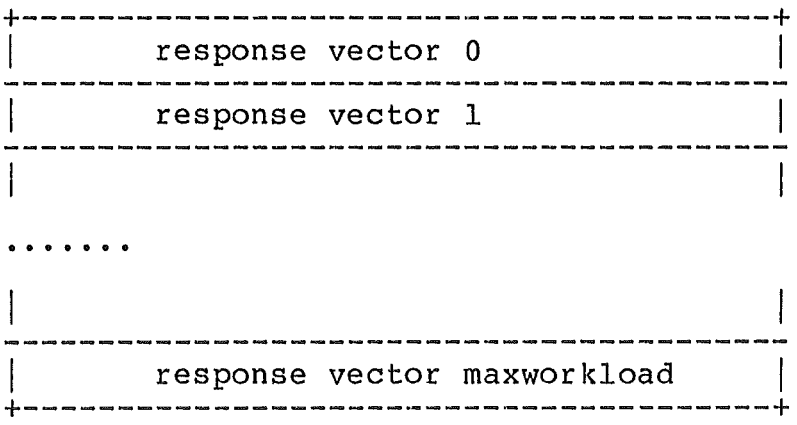
5 PLOTTING COMPONENTS OF RESPONSE TIME

Section 3 described a method for determining the components of response time, and a monitor to perform the breakdown. The remaining problem was to plot these components against workload.

5.1 The response array

The "response array" is useful for storing data which enables plotting of response time components against a measure of workload. Each response array has an associated workload measure (e.g. logged in terminals), which is used to index the array. A response array can be thought of as an

ARRAY [0..maxworkload] OF response vector, i.e.



where maxworkload is an estimate for the maximum observed value of the associated workload measure.

Every time a valid sample is taken, the value of the workload measure is calculated, and the response vector is added into the row of the response array which corresponds to the workload measured. Response vectors for any samples in which the workload measure is greater than maxworkload, are added into the maxworkload row of the response array. The number of response vectors added into each row of the response array is recorded by a count in each response vector.

A response array allows plotting of any function of the counts within a response vector against the selected workload measure. Therefore, plots of response time components against workload are possible using data from a response array.

5.2 Selection of Workload Measures

Selection of appropriate workload measures was the final decision to be made. Criteria considered in making this decision were that :-

- the workload measure should reflect the actual workload on the system, and

- the workload measure should be meaningful.

No single workload measure would satisfy these somewhat conflicting requirements. Therefore, two workload measures were selected :-

(i) Number of logged-in terminals. This measure is meaningful, as logged-in terminals is easily related to the number of people using the system. It is not, however, a particularly accurate reflection of workload. The fact that a terminal is logged-in does not necessarily mean that the terminal is contributing to workload. This measure also ignores any contributions which phantom processes make to the workload. It is, however, essential in evaluations which require some sort of estimate of "the number of terminals supported" by a system.

(ii) Number of active users (= active terminals + active phantoms, where active is as defined in section 3). This measure bears a stronger relationship to the actual load the system is under, and is the best measure to use in comparisons where a relationship to the number of people using the system is unimportant. Active users is not, however, particularly meaningful, as it is not readily related to the number of people using the system.

The state in which the system is found when a sample is taken is influenced by the workload in the immediately preceeding period. This fact must be recognised when determining active users, because of the rapidly changing nature of this workload measure. Therefore, the active users workload measured for a sample is determined by averaging active users found in that particular sample, and in the four preceeding samples (the choice of 5 as the number of samples averaged over is somewhat arbitrary).

The number of logged-in terminals changes slowly, so an averaging technique is not required for this workload measure.

5.3 The UNI monitor

The monitor constructed to maintain response arrays for the two above workload measures is the UNI monitor, the second of the current ASHMON monitors to be discussed. The two response arrays are built up for a specified number of valid samples. When sampling finishes the two response arrays, along with identification, are written to a data file. The information in the data file can be processed by two programs :-

- a program which converts a response array into a performance summary. A total response vector may be derived from a response array by a summation of the rows of the response array.

- the PLOT program. This program can plot many functions of response vector counts against either of the two workload measures selected.

For convenience both response arrays maintained by UNI have the same dimensions.

5.4 PLOT program

The PLOT program is a powerful tool designed to produce graphical information from UNI data files. The use of the PLOT program, and the facilities it offers, are detailed in Appendix 2. The most important features of PLOT are that :-

(a) any of the different types of information available in a performance summary can be plotted (as a function of response vector counts) against the two workload measures. Plots of sample density are also available, and can be used to determine the range of workload values in which plotted points represent significant numbers of samples.

(b) the provision of three Y-axis measures for functions (f(x)) of response vector counts :-
(i) average f(x) per sample.
(ii) f(x) as a percentage of response time.
(iii) f(x) as a percentage of total time.

(c) the ability to generate plots with either of the two workload measures on the X-axis

(d) choice between two different types of plot files. Plots may be written to text files, or to NCAR metafiles. NCAR plots are better quality, but are expensive and slow to produce. The rarity of NCAR graphics packages on Prime sites is also a problem. Text file plots are not as accurate, but are cheap and universally useful.

5.5 SYSPRO - the UNI monitor implemented as an interrupt process

One of the interrupt processes within Primos, the SP1 process, is a spare interrupt process. If the Primos source codes are available an interrupt process can be added as the SP1 process. A version of the UNI monitor has been installed as the SP1 interrupt process, giving the following advantages :-

- the SP1 process runs at a higher priority than the disk processes, so fewer samples are rejected because of disk data discrepancies.

- the code for all interrupt process must be locked in memory,

which means that page faults cannot occur during data gathering, and the monitor therefore causes no disk IO overhead.

- as the SP1 process is a part of the operating system it can access CLKRNG semaphores directly. Therefore the same CLKRNG semaphore can be waited on between samples. This fact, along with the high priority of the SP1 process, means that sampling frequency will be more regular, and data produced more reliable.

- only one monitoring session can be in progress at any time, as there is only one SP1 process. This is desirable as the simultaneous operation of ASHMON monitors can distort the results obtained.

UNI was converted to run as the SP1 process, and the resulting monitor was called SYSPRO. The conversion was reasonably straightforward, with communication between SYSPRO and user programs to start SYSPRO, and to retrieve data from a SYSPRO monitoring session, causing most of the problems.

When idle SYSPRO waits on the SP1SEM semaphore. When this semaphore is notified SYSPRO performs a monitoring session, after which it returns to SP1SEM. SYSPRO's user interface is provided by two user programs :-

(i) A program to start SYSPRO. This program sets the number of samples to be taken by SYSPRO, and starts SYSPRO, provided it is not already in use, by notifying SP1SEM.

(ii) A program to retrieve data from SYSPRO, after a monitoring session. If SYSPRO is in use an error message is given.

To implement SYSPRO the monitor and its utility routines were loaded into Segment 16, and some minor changes were made to the Primos source code. SYSPRO has not interfered with other system operation, and is the most commonly used ASHMON monitor.

SYSPRO data files have the same format as UNI data files, so all data presentation programs available for UNI data files also work for SYSPRO data files. The usage and installation of SYSPRO are described fully in Appendix 2.

5.6 Combination of Data file

ASHMON also has a data file combination program, COMBINE. Monitoring is generally done during one day, so for longer periods combination of data files is necessary.

Any number of data files produced by the same monitor may be combined. Files combined may be original data files, or files produced by previous combinations. Because the SYSPRO and UNI monitors run under different circumstances, SYSPRO data files may not be combined with UNI data files. Full details on the use of the combination program are given in Appandix 2.

6 PORTABILITY

The importance of producing a monitor portable to a large range of Primes was discussed in Section 2. ASHMON has been designed with portability in mind, with all system-dependent information grouped into three insert files :-

(i) PMA constants,

(ii) Pascal constants,

(iii) Pascal initialisations.

To port ASHMON to another Prime installation requires the changing of the data in the three insert files. Initially it was intended that the changes should be made manually, but this task is not particularly easy and there is scope for error.

A program (CONFIG) was therefore written to set up the three insert files. The information required for the insert files is :-

(i) names and physical device numbers for all local disk partitions, including paging partitions.

(ii) the number of counts within the response vector. The number of counts in the response vector depends on the number of local disk partitions.

(iii) logical addresses of semaphores and locks, used for comparisons with addresses found in PCBs.

(iv) the addresses of data structures to be sampled i.e.
- start of PCB and disk QCB areas
- addresses of disk queue headers and device busy bits
- address of user priority 3 on the ready-list

(v) the location of the global clock GCLOK.

(vi) the number of terminal and phantom users that the system is configured for.

(vii) the program counter (PB) of a logged-out terminal process (see 3.2.3).

The following assuptions are made :-

(i) there are 97 numbered semaphores

(ii) there are 10 CLKRNG semaphores

(iii) the length of PCBs and disk QCBs, and the position of data within them, does not vary between systems.

(iv) that PHMSEM, HIPRIQ, ELIGQ, and LOPRIQ are in Segment 4,

and all other semaphores identified in Section 3 are in Segment 6.

(v) that PCBs and the ready-list are in Segment 4.

All of these assumptions should be correct for Rev 19 versions of Primos.

6.1 Gathering data for insert files

The methods used by CONFIG to gather data to construct the three insert files are now described. Data gathering can be divided into a number of parts.

6.1.1 Invariant information

The first step is to put information which does not change between Primes into the insert files.

6.1.2 Information required from the user

When ASHMON is configured, the user is asked for :-

- a name that ASHMON can use to identify the system.

- a value for maxworkload, used to dimension the response arrays. As the number of logged-in terminals is usually higher than the number of active users, it is suggested that the number entered should be slightly above the maximum number of logged-in terminals usually experienced.

- whether the system has an NCAR graphics package.

- whether the load map for the current version of Primos (RING0.MAP) is in the PRIRUN directory. This is the usual situation, but occasionally the operating system files may be in directories other than PRIRUN.

6.1.3 Disk information

CONFIG derives all the disk information that it requires from the names and the physical device numbers of the local disk partitions. Physical device numbers and names are obtained from two sources :-

(i) Physical device numbers and names, for file partitions on local disks, are found by calling the system subroutine LDISK$. LDISK$ does not return this information for paging partitions, however.

(ii) Physical device numbers and names for paging partition(s) are found by searching the file CMDNC0>CONFIG.

6.1.4 Response vector counts

When the number of local physical devices has been determined by the above method, response vector constants are written to the insert files.

6.1.5 Extracting information from RING0.MAP

The system load map (RING0.MAP) contains logical addresses of system entry points. The following data can be extracted from RING0.MAP :-

- addresses of all semaphores,

- address of GCLOK,

- starting addresses for the PCB and QCB areas,

- addresses of data needed during CONFIG. The number of configured terminals and phantoms, and the number of QCBs, are found in this way.

- addresses needed when searching for device busy bits (see 6.1.6). These addresses correspond to the entry points DSKNW_ and LOCKRH.

- address of user priority 3 on the ready-list.

6.1.6 Information Remaining

Three other pieces of data must be determined :-

(1) the address of the disk queue headers
(2) the PB of a logged out terminal
(3) the address of the device busy bits

These address do not appear in RING0.MAP, and procedures for finding them are awkward.

(1) Disk queue headers

In current versions of Primos, the disk queue headers appear immediately after the disk QCBs. This fact can be used to determine the address of the disk queue headers, as the address of the start of the disk QCBs, their number, and their length are all known.

(2) PB of a logged-out terminal process

It is possible to determine whether a terminal is logged-in by a call to GMETR$. To find the PB of a logged-out terminal process, CONFIG steps through the terminal PCBs, recording the PB of processes which GMETR$ shows to be logged-out. CONFIG stops looking when it finds that the PBs in the last three logged-out terminal PCBs are the same. The requirement that the last three be the same greatly reduces

any possibility of error.

(3) Device busy bits

Finding the address of the device busy bits is not easy. They are to be found somewhere between the DSKNW_ and LOCKRH entry points. Virtual memory is searched from DSKNW_ until either the device busy bits are found, or until LOCKRH is reached (in which case a configuration error occurs, see 6.2). CONFIG selects a controller which is present on the system, and searches for its device busy bits As the busy bits for the two controllers are found in adjacent words, only one word must be located in this way.

To decide whether a word holds the device busy bits, it is compared against the disk queue header word for the relevant controller :-

- if the queue header is 0 the word must also be 0.

- if the queue header is non-zero the word must be in the range 1 to 15, as the device busy bits for a busy controller must be in this range.

- the above correspondences must be found for 10 observations. Two chances are given for errors, and at least 2 of the correspondences must occur for a non-zero queue header word. If the last condition were not imposed a word containing 0 could be mistaken for the device busy bits during periods of very light disk activity. If 2 instances of a non-zero queue header word do not occur in the 10 observations, then observations are continued until the 2 non-zero queue header word observations occur, or the word is rejected.

6.2 Summary of CONFIG

CONFIG uses a number of techniques, some of them inelegant, to extract the system information it requires. CONFIG has so far been successfully used to configure ASHMON for 3 different Primes : a P550-II at Mainland Data Processing, a P250-II at Prime Wellington, and the University of Canterbury P750. ASHMON will be put onto three further Prime systems in the near future.

There are a number of possible problems which CONFIG is designed to report on :-

(i) No paging partition found in CMDNC0>CONFIG.

(ii) An address searched for in RING0.MAP is not found.

(iii) The device busy bits are not found.

Errors (ii) and (ii) indicate fundamental problems, which must be solved by someone who is familiar with the ASHMON source programs.

7 VALIDATION

Validation of results is an essential part in the testing of any monitor. With software sampling monitors, there are three important factors in validation :-

(i) That results produced are correct. Results obtained should be verified, where possible, against results obtained by established monitors.

(ii) That sampling is genuinely random, and not synchronised with system events.

(iii) That the monitor operates with acceptable overhead, as the overhead of any software monitor affects the accuracy of results.

7.1 Validation of ASHMON results against USAGE

The Prime software monitor USAGE[6] gives information on CPU, memory, disk, and file system usage, by monitoring the accounting meters. It is possible to compare some of USAGE's results with measures provided in an ASHMON performance summary. There are two areas in which comparisons can be made :-

(i) CPU utilisation, and

(ii) running times on each drive as a % of total disk running times.

Comparisons have been made for results from the UNI monitor, against those of USAGE.

(i) CPU utilisation

To be able to compare CPU utilisations given by USAGE and ASHMON, the utilisations computed should be in the same terms.

USAGE provides a number of CPU utilisation statistics :-

- % CPU utilised by user processes

- % CPU idle

- % CPU utilisation by some of the more important interrupt processes, and by all user processes active over the interval

An ASHMON performance summary includes a count of the number of samples for which terminal processes were "running" on the CPU. Therefore, dividing the CPU running frequency by the number of samples taken gives an estimate of the percentage of CPU time available to user processes which is used by terminal processes. The reason that the denominator is the time available to user processes and not total time, is that the UNI monitor can only monitor the CPU use of processes of

lower priority than itself.

It is possible to get an estimate of the ASHMON CPU utilisation from USAGE data in the following way.

%CPU available to user processes of priority less than 3 =
%CPU used by all user processes -
%CPU used by user processes of priority of, or less than, 3 +
%CPU idle.

User processes which run at a priority above 3 are the Console and NETMAN, the network server phantom, which both run on the "Console" process priority.

CPU time used by terminal processes can be calculated by subtracting the sum of the CPU time used by non-terminal user processes from the CPU time used by all user processes.

Dividing the CPU available to user processes of priority less than < 3 by CPU used by terminal processes gives a USAGE utilisation comparable to the estimate given by ASHMON. USAGE gives a signed error estimate, which can be added to CPU utilisation for user processes to provide a second USAGE figure.

(2) Disk IO activity times

This comparison is much easier to make. USAGE in its disk IO statistics gives the % of total disk activity time attributable to each drive. These percentages may be estimated from an ASHMON performance summary by totalling the running frequencies for all drives, and calculating the percentages of this figure that each drive accounts for.

A validation of ASHMON by USAGE

The ASHMON UNI monitor and USAGE were run concurrently for 20,000 seconds on 4 consecutive days (Tues-Fri). The results obtained are as follows.

% user CPU available used by terminals

Day	USAGE	USAGE with error	ASHMON
Tuesday	74.3	72.4	75.9
Wednesday	67.9	66.3	69.9
Thursday	69.3	67.6	71.5
Friday	56.3	55.2	59.4

% disk run on each drive

Day	Monitor	Cont 0 Drive 0	Cont 1 Drive 0	Cont 1 Drive 1
Tuesday	USAGE	65.00	33.97	1.03
	ASHMON	65.74	33.17	1.08
Wednesday	USAGE	56.69	41.98	1.33
	ASHMON	55.54	42.98	1.48
Thursday	USAGE	67.72	31.62	0.66
	ASHMON	66.73	32.57	0.70
Friday	USAGE	67.00	30.54	2.46
	ASHMON	66.84	30.87	2.29

This validation is fairly informal, with no check of the statistical significance of the difference between the variables compared. It does give a strong indication, however, that ASHMON monitors do give results which are fairly close to those given by USAGE. USAGE is a monitor which has been in use for several years and should be accurate enough to use to validate ASHMON. Disk and CPU related measures are the most likely ASHMON statistics to be in error (because of the complex way in which they are estimated), so their validation is of most significance.

ASHMON CPU figures are fairly accurate, but are consistently higher than USAGE CPU figures. The reason for this is discussed in 7.2. The disk statistics are very close to those provided by USAGE, with no obvious pattern to the small fluctuations that occur. Overall ASHMON is well validated by this accuracy test.

7.2 Coincidence with regular events

All ASHMON monitors wait on CLKRNG semaphores between samples, as this is the is the only mechanism Primos provides for timed suspensions of processes. (CLKRNG and its use are outlined in Appendix 1, Section 4). The mechanism is a very coarse way of suspending processes, and causes event synchronisation problems for ASHMON monitors.

CLKRNG contains 10 semaphores, so on average 10% of the processes which are on CLKRNG will be moved to the ready-list each time a CLKRNG semaphore is drained by the CLOCK process. When the process running an ASHMON monitor is moved from a CLKRNG semaphore onto the ready-list it will find all of the user processes that were on the same CLKRNG semaphore, on the ready-list. This causes the CPU running and ready-list frequencies to be eexaggerated, and the CLKRNG frequency to be underestimated, by a little under 1/9 of the CLKRNG frequency recorded. ASHMON's higher CPU figures in 7.1 are caused by the method Primos uses for suspending processes for periods of time.

A solution to this problem would be to alter the CLOCK process so

that it notifies a special MONITOR semaphore every second, at a time which does not correspond to any existing timer interrupts (as far as is possible). This could have been done at Canterbury, as the source listings for Primos are available, but it would have made ASHMON much less portable.

A further synchronisation was noted for SYSPRO. Originally SYSPRO waited on the first CLKRNG semaphore between samples. Results showed exceedingly low CPU running frequencies. The problem proved to be that the NETMAN process (a phantom process at a priority above user priority 3) was woken up at the same time the first CLKRNG semaphore was notified, and was the running process every time SYSPRO sampled. To solve this problem, SYSPRO now waits on the 7th CLKRNG semaphore, and starts its ready-list scan at user priority 3, avoiding NETMAN, and so being consistent with the other ASHMON monitors.

7.3 Overhead

A software monitor should make minimal use of the system monitored, to ensure accurate results. Resource utilisations for the ASHMON UNI monitor can be determined from USAGE. Overhead figures for this validation were drawn from the USAGE outputs used in 7.1. There are three overheads of interest :- %CPU used, %IO used, and memory requirements.

ASHMON overheads

Day	%CPU	%IO	Mem pages
Tuesday	0.724	0.032	11
Wednesday	0.716	0.022	10
Thursday	0.716	0.016	10
Friday	0.702	0.008	10

Examples of normal overheads are given in [3]. In a survey of measurement tools 19 claimed to have a CPU overhead of less than 1% of CPU time, with the highest figure at 6%. Memory for monitors on IBM machines averaged 13Kb. Buzen, also in [3], suggested a figure of 6-12Kb memory required, and a typical CPU overhead of 1%-5% (although overheads of 40% were not unknown).

ASHMON CPU and IO overheads are obviously acceptably low (figures above are for UNI). UNI needs around 10 pages of memory (20Kb) which may seem a little large when compared to the figures quoted by Buzen. However, his figures are from 1977 and memories are now larger and much cheaper. The University of Canterbury P750 [8] has 4Mb of memory, of which Primos requires about 1Mb. UNI therefore requires about 0.6% of memory available to users, which is acceptably low.

ASHMON is designed to run on any 50 series machine, apart from the P850. Overhead will be greater for the smaller machines of the series. The least powerful of the Primes currently produced is the P250-II, whose CPU is about 15% as powerful as the P750. The slower processor speed would be compensated for, to some degree, by the smaller amounts

of data to be collected for the smaller machine. (Overhead depends on the number of user processes, and on the value specified for maxworkload). A P250-II could be expected to have 2Mb of memory, enough to make UNI needs insignificant. IO needs will remain minimal. Some preliminary measurements made on a P250-II at Prime Wellington indicated acceptable monitoring overhead.

The preceeding discussion applies to the UNI monitor. The SYSPRO monitor is very similar to the UNI monitor. It has a comparable CPU overhead, needs no IO as it is wired into memory, and needs a little less memory. The RR monitor is much smaller than the UNI monitor, and needs less of all three resources.

7.2 Summary of Validation

The ASHMON monitors have been substantially validated, as shown by the results quoted in this section. Some distortions are caused by the crude Primos timer mechanism which results in overstated CPU running and ready-list frequencies, and understated timer frequencies. USAGE has shown ASHMON monitors to be fairly accurate, and that distortions are not great. Overhead measured was very low for the P750, and indications are that overhead would be acceptable over the entire 50 series range.

8 RESULTS

This report has focussed on the design and development of ASHMON, rather than on any performance evaluation studies undertaken with ASHMON. This section is included to give examples of results which ASHMON can produce, and the information that can be taken from them.

Many examples could have been given, but this brief illustration is restricted to results gathered on Monday 20 August by SYSPRO for the University of Canterbury P750. On this day the Prime had what was probably its greatest interactive workload, and worst response, for 1984. During the 20,000s (approx 6 hour) period monitored, the number of logged-in terminals ranged from 19 to 39.

A performance summary and four plots have been prepared from the data gathered, and these are discussed in the following subsections.

8.1 Performance Summary

The performance summary gives a reasonable overview of system performance. Some of the more interesting points from Table 1 are :-

(i) in the response time breakdown, the CPU forms the major component of response time, indicating that it is almost certainly the bottleneck. (Confirmation can be found in the graphical output). This has been the experience at Canterbury over recent months. An interesting feature is that despite the CPU being the overwhelming factor in response time, terminals were found to be running on the CPU for only 12,831 samples out of a total of 20,000. i.e. 64% of CPU time available to user processes was utilised by terminals. The probable cause of this is phantom activity, as on average 4.5 phantoms were active on each sample.

(ii) The controller 0 component of resposne time is approximately three times greater than that for contoller 1, although the ratio of transfers will be somewhat less. This indicates some imbalance of disk traffic between the two controllers.

(iii) response time is 33% of total time. This means that, on average, a terminal user was spending 1/3 of his session waiting for system response, and indicates poor system responsiveness.

(iv) the detailed disk traffic analysis suggests that a large proportion of disk traffic was due to paging, that is transfers involving PAGDEV and ALTDEV. There is a possibility that, although disk traffic is on average a small component of response time, under high workloads thrashing occurs, and is the major component of response time. This suspicion may be checked by examination of the appropriate plots. Another possible reason for the high proportion of paging transfers found is that most interactions in the period sampled were not particularly IO intensive.

(v) user counts are all very high for the P750. An average of 14.5 active users is much higher than measured at other times.

(vi) an average response ratio of 11.49 represents abysmal response time performance, especially considering that errors in calculation are likely to underestimate response ratio. While response ratio would have been better in some periods, in others it would have exceeded 20!

The above set of observations indicate a day with particularly poor response, almost certainly due to a CPU bottleneck. These results are for one day only, with a workload which contained a large number of COSC302 students working on Pascal assignments, resulting in a somewhat atypical workload. This summary is, however, similar to most summaries gathered on the P750 this year.

8.2 Interpretation of graphical output

Four figures have been included as examples of output possible from the ASHMON plotting facility. All figures contain vertical lines showing the 5th and 95th percentiles of sample frequency. The percentile information was drawn from plots of cumulative frequency, which are not included in this report.

8.2.1 Figure 1

Figure 1 shows the relationship between the number of logged-in terminals and response ratio. Response ratio is above 6 for all points plotted, which indicates bad response. To be able to detect an elbow in the curve (and hence estimate the number of logged-in terminals supported by the system) more points would be required for lower numbers of logged-in terminals.

The nature of the curve suggests that a relationship does exist between logged-in terminals and workload, but that the relationship is a little weak. This confirms what was said in 5.2 about the nature of logged-in terminals as a workload measure.

8.2.2 Figures 2 and 3

Figures 2 and 3 are both plots of major response time components against active users, but with different Y-axes. Figure 2 plots response time components as the average number of terminals in each component, and Figure 3 plots the components of response time as a percentage of response time. Figure 2 shows the absolute size of response time components, and Figure 3 shows the relative size of response time components.

One thing illustrated clearly by both plots is that CPU is the major component of response time at all levels of workload, with no dramatic increase in the disk component at high workload levels. This indicates that there is not a thrashing problem, and that the CPU is the bottleneck.

8.2.3 Figure 4

Figure 4 shows the major components of total time, plotted as a percentage of total time, against logged-in terminals. As one would expect, the percentage of time a terminal user spends waiting on the system increases as the number of logged-in terminals increases. This is well illustrated in Figure 4.

8.3 Summary

The results selected for inclusion in this report give some insight into performance on the University of Canterbury P750, and into the capabilities of ASHMON. The examples selected are a very small number of interesting results found with ASHMON, for the Canterbury and MDP Prime systems, over the last few months.

9 CONCLUSIONS

The principal design goal of ASHMON was to produce monitors capable of plotting response time components against workload. A package has been developed containing the following programs :-

- an interactive monitor for estimating response ratio.

- a monitor to gather data suitable for plotting, available in versions which run either as a user process, or as the SP1 process.

- programs to produce plots and performance summaries from data gathered by monitors.

- a program to combine files of data collected during different monitoring sessions.

- a program to set up the system dependent insert files. This feature makes ASHMON portable to any Prime 50 series machine other than the P850.

- an overall control program which gives access to all of the above ASHMON programs (see Appendix 2).

Two modifications to the package would probably be worthwhile :-

(i) identifying processes on the ready-list which have just come off the CLKRNG semaphore that the ASHMON monitor was on. Such processes would be classified as waiting on CLKRNG, rather than on the ready-list. This would overcorrect the current bias toward CPU running and ready-list frequencies, but the amount of the error would be much smaller than it is at present.

(ii) Future developments could include a monitor which builds up response vectors for individual terminals or groups of terminals. A specialisation of such a tool, with a higher sampling frequency, could be used as a monitor for evaluating program performance.

The second modification would be relatively easy, and would mean that ASHMON could duplicate a large number of the performance monitoring functions of Prime's chief monitor USAGE. ASHMON provides many types of information not available from USAGE, or Prime's event detection monitor, GEM.

- components of response time, which can be plotted against workload.

- estimation of response ratio.

- measurements of interactive workload.

It is therefore suggested that ASHMON is a much more powerful monitor than USAGE or GEM, and fills a large gap in Prime monitoring software. ASHMON has the unusual feature of being portable to a range of Prime machines, while not forming part of the operating system.

The development of ASHMON has produced some concepts (and perhaps data manipulation software) which could be exploited on machines other than Primes. The ASHMON concepts of response vectors and response arrays could conceivably be transported to machines of a similar kind. Data manipulation programs written for ASHMON response vectors and arrays could easily be ported to such machines, as the nature of the processing involved is essentially the same, the difference being in the contents of the response vector. These aspects of ASHMON give a small degree of system-independence, not present with other performance monitors.

References

[1] "Primon - a performance monitor for Prime Computers",
A McGregor,
Masters Thesis, Massey University.

[2] "Measurement and Tuning of Computer Systems",
D Ferrari, G Serrazi, A Zeigner,
Prentice-Hall, 1983.

[3] "Infotech State of the Art Report on System Tuning",
Infotech International, 1977.

[4] "Tools and Techniques for Effective Analysis - ,
Computer Performance Evaluation",
M F Morris, P F Roth,
Van Nostrand Reinheld Co, 1982.

[5] "A Methodology for Interactive Service Measurement",
M D Abrams, P Treu,
CACM December 1977, Vol 20, Number 12.

[6] "Prime Systems Operators Guide",
Prime 1982.

[7] "The Systems Architecture Guide Rev 19.0",
Prime 1982.

[8] "Capacity Planning for Large Minicomputers",
I Loffler,
Journal of Capacity Management Vol 1, Number 1, 1982.

[9] "Performance Criteria, Measurement and Analysis
for Interactive Systems",
G C Barney, G C Pentzaropoulos, W Swindells,
Computer Performance, Vol 3, Number 1, 1982.

[10] "Measurement and description of time-sharing response",
J P Penny, P J Ashton,
Computer Performance, Vol 5, Number 3, (to appear).

TABLE 1

Performance summary

1. Identification

System sampled :- University of Canterbury P750
Date of first sampling file
  Date of sample :- 20 AUG 84 Time sampling finished :- 16:24:32
Date of final sampling file
  Date of sample :- 20 AUG 84 Time sampling finished :- 16:24:32
Number of files combined :- 1
Monitor used :- SYSPRO
Samples taken :- 20000
Samples discarded through inconsistent data :- 101
Samples discarded through P/F during sample :- 0

2. Performance results

Response time

CPU	173832	87.5	Running 12831			
			Queueing 161001		->Ready-list	37792
					HIPRIQ	15633
					ELIGQ	42205
					LOPRIQ	65371
Controller 0	15038	7.6	Running	6917		
			Queueing	8121		
Controller 1	5513	2.8	Running	3878		
			Queueing	1635		
Waits QCB's	0	0.0				
Other peripherals:						
Mag tape	446	0.2				
Nl locks	3551	1.8				
Network waits	0	0.0				
Others	313	0.2				
		100.0				
Total response time	198693	33.1				

Input

Think and input time	384569	64.0

Output

Waits for term buffer	17406	2.9

```
 Others
 ------
   Waits on user sems        423     0.1
                            -----   -----
Total time                 601091   100.0

 Further analysis of disk traffic

 Controller 0

  Disk 0    Running  :-  6917    46.0
            Queueing :-  8121    54.0
   CANT00         Running  :-  1435     9.5
                  Queueing :-  1927    12.8
   PAGDEV         Running  :-  3594    23.9
                  Queueing :-  4261    28.3
   CANT01         Running  :-  1888    12.6
                  Queueing :-  1933    12.9

 Controller 1

  Disk 0    Running  :-  3765    68.3
            Queueing :-  1631    29.6
   CANT02         Running  :-   475     8.8
                  Queueing :-   268     5.0
   ALTDEV         Running  :-  2743    50.8
                  Queueing :-  1153    21.4
   CANT03         Running  :-   547    10.1
                  Queueing :-   210     3.9

  Disk 1    Running  :-   113     2.0
            Queueing :-     4     0.1
   CANT04         Running  :-   113    96.6
                  Queueing :-     4     3.4

Average number of logged in users :-  48.0
   Average number of terminals logged in :-       30.1
   Average number of non-terminals logged in :-   18.0
Average number of active users :-  14.5
   Average number of active terminals :-        9.9
   Average number of active non-terminals :-    4.5

Mean Response ratio = 11.49
```

SAMPLED — UNIVERSITY OF CANTERBURY P750 FIRST DATE 20 A AT 16 24 3
FINAL DATE 20 AUG 84 AT 16 24 3 SAMPLES 20000 DISK TS 101
PAGE REJECTS 0 FILES — 1 MONITOR — RO

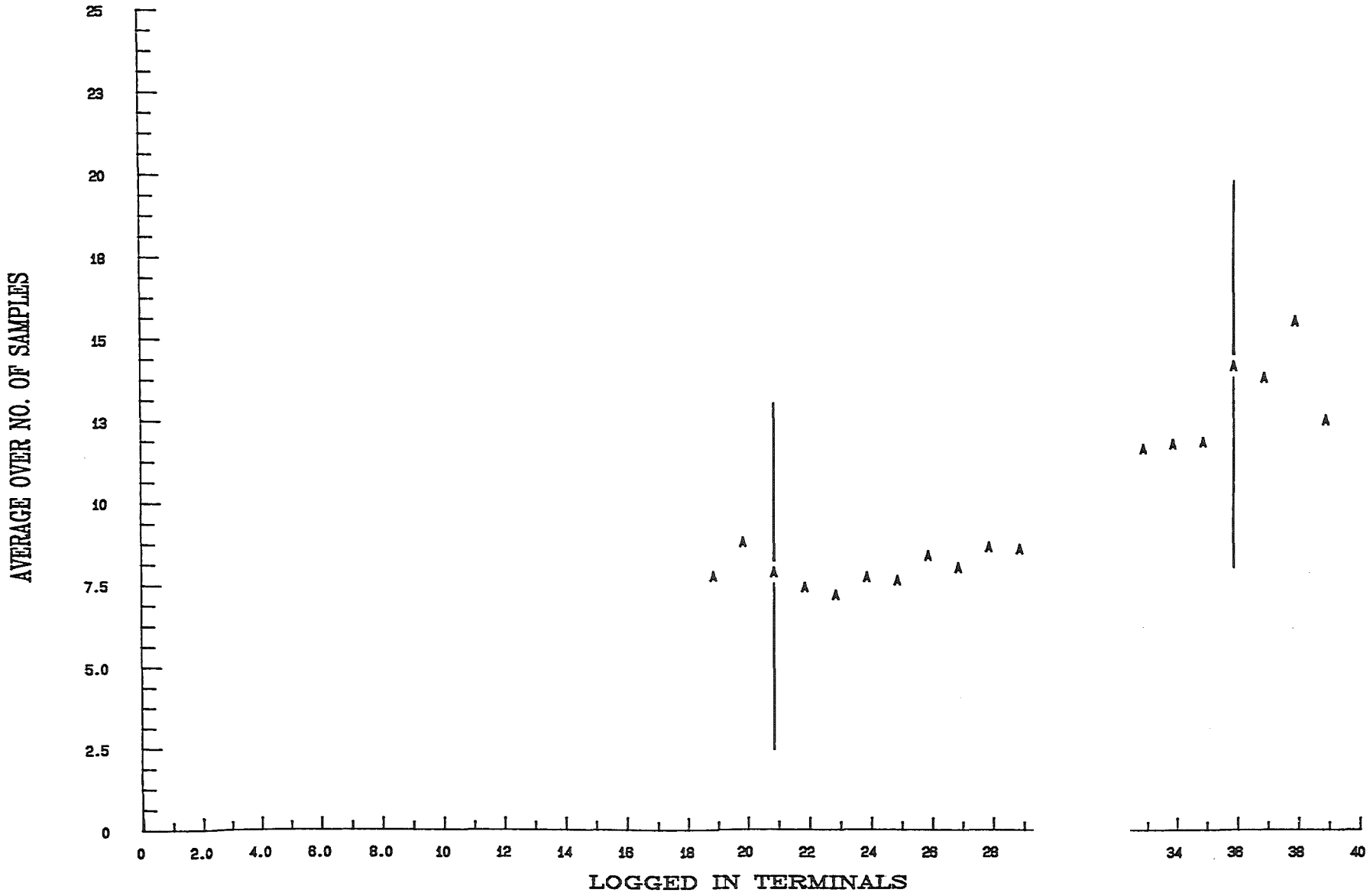


Figure 1

SAMPLED — UNIVERSITY OF CANTERBURY P750 FIRST DATE 20 AU AT 16 24 3
FINAL DATE 20 AUG 84 AT 16 24 3 SAMPLES 20000 DISK 'S 101
PAGE REJECTS 0 FILES — 1 MONITOR — 0

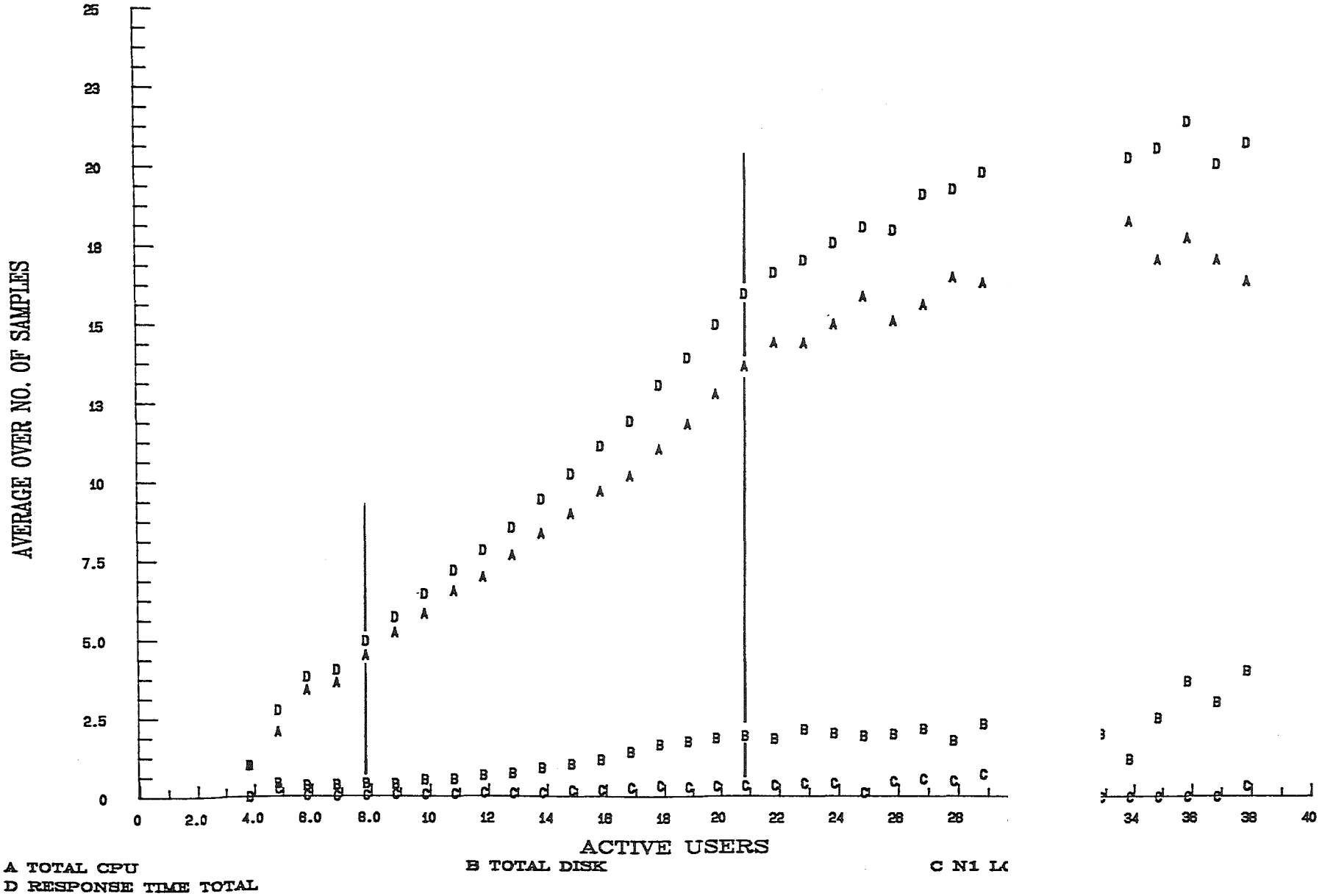


Figure 2

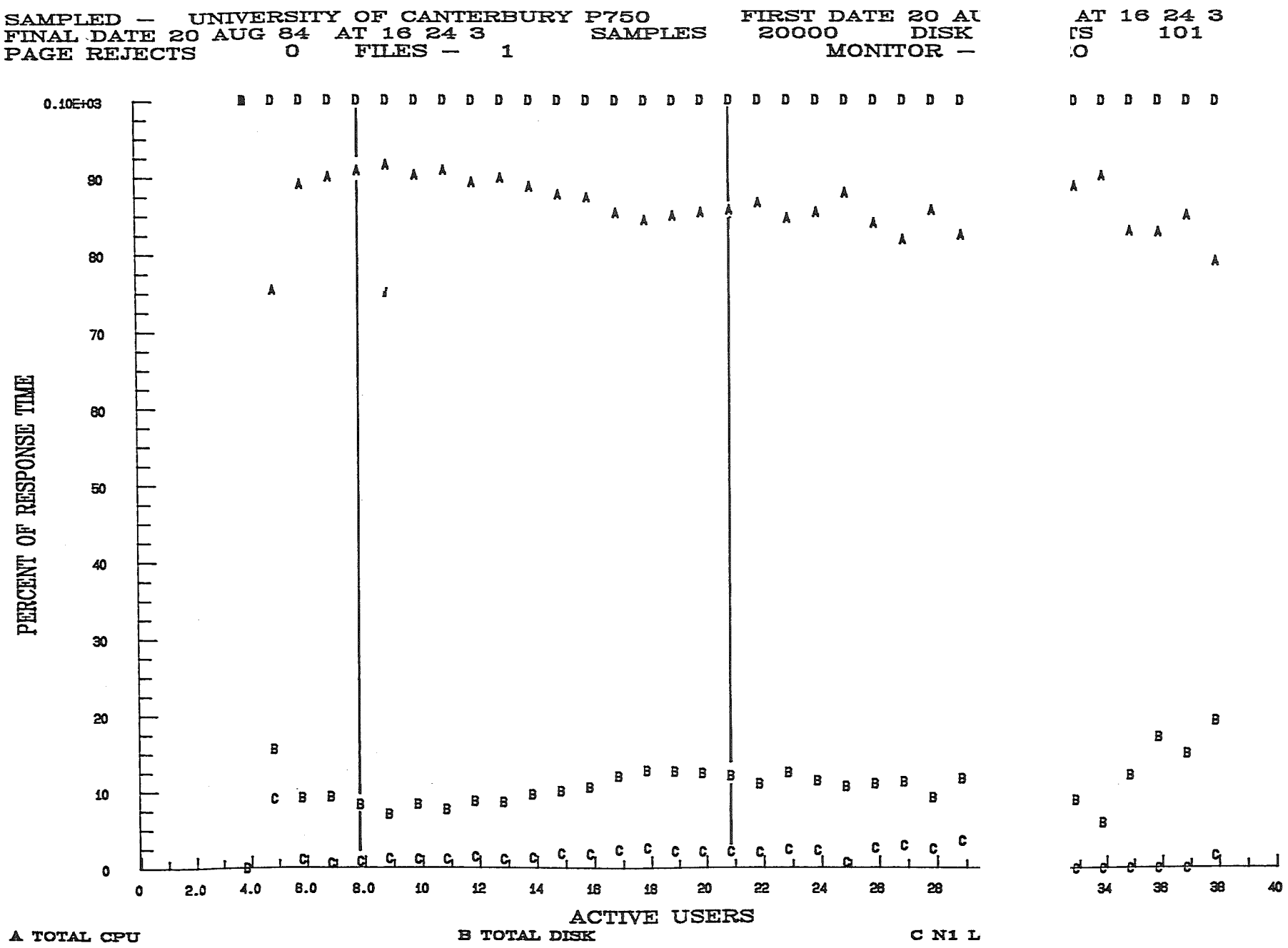

SAMPLED — UNIVERSITY OF CANTERBURY P750 FIRST DATE 20 AU AT 16 24 3
FINAL DATE 20 AUG 84 AT 16 24 3 SAMPLES 20000 DISK TS 101
PAGE REJECTS 0 FILES — 1 MONITOR — O
0.10E+03
90
80
70
60
50
40
30
20
10
0
PERCENT OF RESPONSE TIME
0 2.0 4.0 6.0 8.0 10 12 14 16 18 20 22 24 26 28
34 36 38 40
ACTIVE USERS
A TOTAL CPU
B TOTAL DISK
C N1 L
D RESPONSE TIME TOTAL

Figure 3

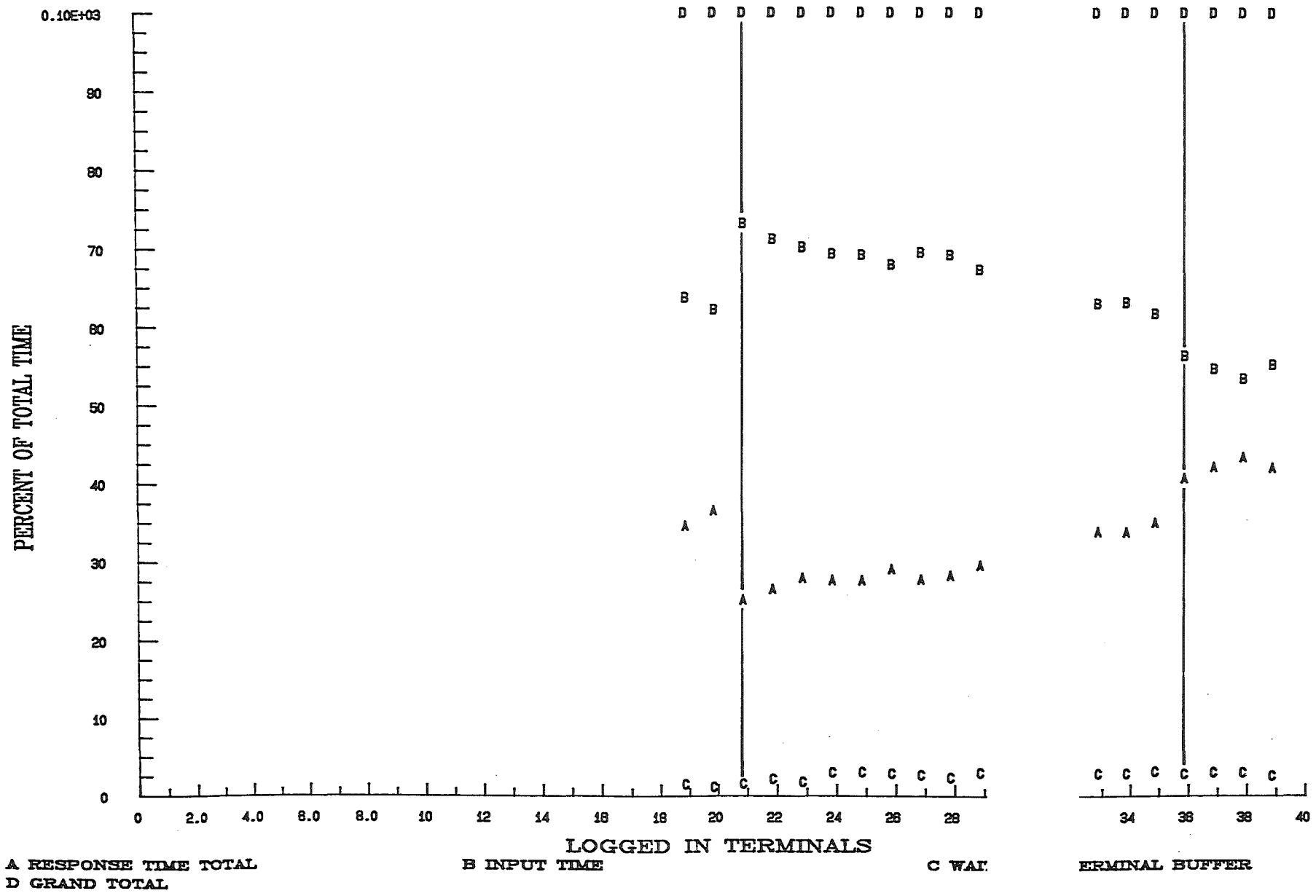

SAMPLED — UNIVERSITY OF CANTERBURY P750 FIRST DATE 20 A AT 16 24 3
FINAL DATE 20 AUG 84 AT 16 24 3 SAMPLES 20000 DISK CTS 101
PAGE REJECTS 0 FILES — 1 MONITOR — RO
0.10E+03
90
80
70
60
50
40
30
20
10
0
PERCENT OF TOTAL TIME
0 2.0 4.0 6.0 8.0 10 12 14 16 18 20 22 24 26 28
34 36 38 40
LOGGED IN TERMINALS
A RESPONSE TIME TOTAL
B INPUT TIME
C WAI
ERMINAL BUFFER
D GRAND TOTAL

Figure 4

Appendix 1 - Background on Prime architecture and Primos

A brief summary is given of those aspects of the Primos IV operating system, and the Prime 50 series architecture, which are relevant to this project. A close look is taken at process scheduling, disk IO and timed delays, all of which are important to this project. Further information on many of the topics discussed may be found in [7].

1. OVERVIEW

The Prime 50 series machines under Primos IV are time sharing systems with a paged, virtual memory architecture. They are "superminis" with a 16-bit word size. Time sharing takes place between operating system (interrupt) processes and user processes. There are a total of 19 interrupt processes (many of them redundant on most systems) which perform functions such as the clock process (responsible for various timer interrupts), interacting with device controllers, and the backstop process (described below).

The number of user processes is an operating system parameter and typically varies from around 20 to in excess of 127, depending on the size and the uses of the system. User processes are subdivided into the categories:

```
System Console :- User 1
Terminals :-      Users 2..nterms
Phantoms :-       Users nterms+1..nusers

where  nterms = number of terminal processes configured,
       nusers = number of user processes configured.
```

Most phantoms are user processes which take commands from a command file rather than a terminal. Some phantoms are used by PRIMOS (logged-in under the SYSTEM user code) for peripheral control (printers, card readers, etc), and network communications.

The Prime also supports a crude batch system which is implemented via a SYSTEM phantom, but most users do batch work by setting up their own phantoms.

Every process (interrupt and user) has a Process Control Block (PCB) which stores information on the state of the process. Types of information stored in each PCB are :-

- priority of process,

- link to next PCB (for lists PCBs waiting on a semaphore, or on the ready-list),

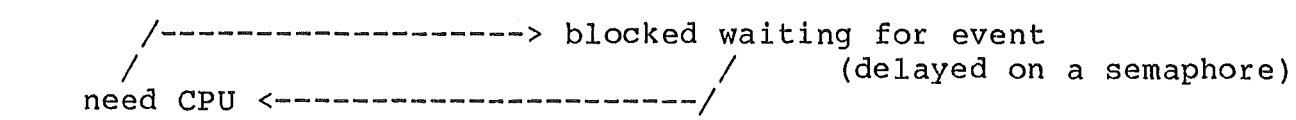
- event being waited on,

- accounting information,

- copies of register contents,

- information used when processing faults.

The lifecycle of a process is of the form:



The event being waited on is indicated by 2 words in the PCB whicl hold either the address of the semaphore the process is waiting on, o indicates that the process is on the ready-list. For example, durin "think time", when the process is waiting for terminal input, the PC is attached to the terminal buffer semaphore.

When a process has to wait for an event (completion of a dis transfer, entering an area of mutual exclusion, etc) it performs a WAI on the relevant semaphore. If the event has already occurred, th process continues. Otherwise, the running process is suspended and it PCB is put on a queue of processes waiting on that semaphore.

When an event occurs, the ultimate effect is a NOTIFY operation o the relevant semaphore. If there are processes waiting on th semaphore then one is removed and put on the ready-list. If th priority of the notified process is higher than that of the notifyin process then the notified process will get the CPU. Otherwise th notifying process continues.

2 SCHEDULING

2.1 Ready-list

Process scheduling on the Prime is controlled by a firmwar dispatcher. A process exchange occurs when a process is delayed on semaphore, a process' time slice expires, or an event external to th process causes an interrupt. The dispatcher then gives the CPU to th process of highest priority of those currently on the ready-list.

The ready-list is an array of PCB queues. Each element in the array consists of pointers to the first and last PCB in that particular list, and represents a different priority level. There are a total of 14 priority levels which are, in decreasing order of priority :-

1 CLOCK
2 AMLC
3 SMLC
4 Multiplexor controllers
5 Versatec contoller
6 Ring net contoller
7 SP1 process
8 Disk controllers
9 Console, NETMAN
10 User priority 3
11 User priority 2
12 User priority 1 (normal user priority)
13 User priority 0
14 Backstop

The backstop process is always on the ready-list, so there is always a process ready to run.

2.2 Backstop, HIPRIQ, ELIGQ, LOPRIQ and MAXSCH

Interrupt processes are always found on the ready-list if they are waiting to use the CPU, but user processes wait on of a group of semaphores before being put on the ready-list. There are two reasons for the existence of these semaphores :-

(i) to enable thrashing control,

(ii) to try to give better service to interactions which require small amounts of CPU time.

The group of semaphores is divided into 3 categories :-

HIPRIQ
ELIGQ
LOPRIQ

HIPRIQ and ELIGQ are single semaphores, which user processes may wait on regardless of priority. LOPRIQ consists of 5 semaphores - one for each of the user priority levels, and one for the Console process priority level (levels 9 to 13 on the ready-list).

Processes are transferred from these semaphores to the ready-list by the backstop process, which runs only when there are no other processes on the ready-list.

If there are any processes on the HIPRIQ, backstop transfers a process to the ready-list by notifying the HIPRIQ sempahore. If HIPRIQ is empty, the two lower priority queues are then considered. Processes are taken from ELIGQ or LOPRIQ only if the number of processes

currently in the system is < MAXSCH. MAXSCH is an operating system parameter used to restrict the number of processes able to use system resources, and gives a simple means of thrashing control.

If the MAXSCH test succeeds, backstop looks at the ELIGQ for a process to run. If the ELIGQ is also empty a LOPRIQ semaphore is examined. The lower priority semaphores in LOPRIQ are examined less often than the higher priority semaphores in LOPRIQ. If there is nothing to run, backstop just keeps monitoring the semaphores until it finds something to run.

When an interaction begins (i.e. input from command file or terminal is given) the process is put onto the HIPRIQ. Backstop moves the process onto the ready-list and processing begins. The process is timesliced after it has used a minor timeslice (1/3 of a second) and is put into ELIGQ. The process is returned to the ELIGQ after a further minor timeslice and is put into the appropriate LOPRIQ after a third minor timeslice. Thereafter the process will alternate between LOPRIQ and ELIGQ, receiving three minor timeslices on each, until the interaction is complete (i.e. next input request).

This strategy is designed to give better service to interactions which require little CPU time.

3 DISK IO

The Prime 50 series machines may have up to 2 disk controllers, each controlling up to 4 drives. Drives are generally 300Mb, 1040-word module drives. Occasionally, 30Mb fixed head drives are used as paging devices. Drives are divided into 1 or more logical areas (or partitions). One or two logical areas (of 30Mb) are set aside as paging areas.

A process wanting a disk transfer calls a routine in DISKIO with details of the disk request. These details are put into one of 17 (in Primos Rev 19) disk Queue Control Blocks (QCB). If no QCBs are available the process waits on a semaphore until one becomes available. QCBs contain :-

- a link to another QCB
- a semaphore
- logical area / cylinder / track / record # of the required record
- parameters to and from disk driver
- error codes.

The QCB, after being initialised, is put into a linked list of QCBs waiting for transfers involving the controller, which controls the drive, that contains the logical area referenced by the transfer. Three lists are maintained within the QCB's - one for pending requests for each controller, and one for the free list. After the QCB has been linked into the appropriate controller list, the process waits on the semaphore within the QCB, which is subsequently notified by a disk controller process when the transfer is complete. Finally, the QCB is

returned to the free list.

Seeks may be overlapped with seeks or transfers on drives connecte to the same controller. To keep a record of which drives are busy, an which are idle, each controller has a word allocated to it for it device busy bits. The 4 low order bits of each of these word specifies whether drives 0..3 respectively are busy or idle.

Only user processes can request disk IO.

4 TIMERS

All suspensions of user or interrupt processes for specified tim periods are done by calling the routine STIMER with a parameter whic specifies the length of the delay in tenths of seconds. The callin process then "sleeps" for at least the specified period.

To achieve the delay, STIMER uses a ring of 10 semaphores calle CLKRNG. Every 1/10 of a second, the CLOCK process notifies al processes waiting on the "current" CLKRNG semaphore (specified b CLKPTR), and steps CLKPTR to the next semaphore in the ring.

STIMER delays a process by recording the clock time at the start o the delay, and then waiting on CLKRNG semaphores until the require time has elapsed. The particular semaphore for each wait is determine by the delay remaining, and the current position of CLKPTR. Obviousl the maximum delay obtainable by one wait is 1 second.

For example if a process wants to "sleep" for 5/10 of a second i will be put onto the CLKRNG semaphore that is 5 positions ahead o CLKPTR.

Appendix 2

ASHMON: A Monitoring Package for Prime 50 Series Machines

1. INTRODUCTION

ASHMON was developed by P.J. Ashton in the Department of Computer Science, University of Canterbury, Christchurch, New Zealand. The monitors in the package are based on an initial design by J.P. Penny. The present version of ASHMON is intended for use under Primos Rev 19.

The goal has been to produce software monitors which allow decomposition of response time into components, and to see how these vary with increasing workload. The monitoring technique involves sampling of various words in the Process Control Blocks (PCB's), and the Disk Queue Control Blocks (QCB's) at 1 second intervals.

The package consists of:

(i) An overall control program (ASHMON) which offers access to all other programs in the package.

(ii) RR - a small interactive monitor designed to estimate the response ratio for interactions in a specified interval.

(iii) A batch monitor able to give a breakdown of response times into components, against a variety of counts of users. The monitor is available in two forms :-

(a) UNI, which runs as a user process, and

(b) SYSPRO, which is loaded into Primos, and runs as the system process SP1.

SYSPRO should give the more accurate results.

(iv) COMBINE, a utility for combining output files from separate monitoring runs.

(v) Two programs for presentation of measurement data:

(a) TORTCOMP presents a performance report, and

(b) PLOT provides graphs of response time components against workload.

The monitoring programs are written in 32-I PMA, with the remainder of the package written in Prime Pascal.

The ASHMON package should be put into a top-level UFD called MONITOR. Before configuring ASHMON, MONITOR>SHARE.SEGS.CPL should be

run from the console to share segments 4 and 6 for reading. ASHMON needs to be able to read from both of these segments. MONITOR also contains a CPL program (MONITOR.BUILD.CPL) which, when run, will run a program CONFIG to set up the system-dependent insert files required by the package. This configuration procedure should be repeated each time there is a system change, for example reconfiguration of logical disks, changes in the number of terminal and/or phantom, users, operating system update.

Appendix C gives an overview of the layout of files within the MONITOR directory and would prove useful if the source code needed to be examined.

2. CONFIGURING THE PACKAGE

2.1 ASHMON

The user code from which the ASHMON package is configured MUST have read access rights to the files CMDNC0>CONFIG and to RING0.MAP.

MONITOR.BUILD.CPL intially compiles and links the overall control program (MONITOR>ASHMON) and the configuration program (MONITOR>CONFIG>CONFIG). Once these two programs have been linked they should not need recompilation as they are largely system-independent. MONITOR.BUILD.CPL then runs the CONFIG program to create the insert files for the rest of the package. The CONFIG program asks the user for:

(i) A name for the system. This is an (up to 32 character) string which serves to identify the Prime that the package is being installed on.

(ii) Maximum number of logged-in terminals. An integer should be given which is a little above the maximum number of logged-in terminals usually experienced (and is preferably a multiple of 10).

(iii) Whether or not (Y/N) the system has an NCAR graphics package. The ASHMON package was implemented on a P750 at the University of Canterbury and interfaces to an NCAR graphics package. The implementation of NCAR at UoC may not be entirely compatible with other implementations and this should be taken into account if NCAR compatibility problems occur.

(iv) Whether or not the system map file (RING0.MAP) is in the PRIRUN directory. This will generally be true, and if so Y should be entered in response to this query. If RING0.MAP is not in PRIRUN then N should be entered, and the full path name of the map file will be prompted for until a valid pathname is entered.

The configuration program then sets up the required insert files, printing various messages as it goes. If any error messages are given

by the CONFIG program then consult P J Ashton, as this indicates a major configuration problem. The final message given by the configuration program is 'Discovery commenced'. This phase of the program involves searching for some pieces of disk information. If disk activity is heavy, Discovery will be brief. If disk activity is VERY light Discovery may take up to 45 minutes.

When CONFIG has finished all remaining programs in the package are compiled and linked.

Any subsequent reconfigurations may be done from the overall control program. You are now in a position to start using the monitoring package.

2.2 SYSPRO monitor

SYSPRO is set up to run as the SP1 process. If you have already set up an SP1 process you may not be able to use SYSPRO. Otherwise, if you have the source code for Primos, you can load the programs to implement SYSPRO into the system. Doing this is a relatively simple procedure which involves the following steps:

(i) Change the definitions for SP1PB, SP1LB and SP1SB in SEG4.PMA to EXT (the corresponding ENT declarations appear in MONITOR>SYSPRO>HIP.PMA)

(ii) Declare the following to be gates:

```
FLGHAN
FIN
WIRPRO
```

(iii) Load these BIN files, in this order, into the same O/S segment (all of these files appear in MONITOR>SYSPRO)

```
WIRPRO.BIN
FLGHAN.BIN
FIN.BIN
ST16.BIN
HIP.BIN
END16.BIN
```

At Canterbury, SYSPRO was loaded into Segment 16, and has proved to be a reliable monitor.

3. USING THE MONITORING PROGRAMS

3.1 Requirements

(i) The UNI and RR monitors are run as user processes, and SYSPRO as an interrupt process. Overlapping use of monitors causes distortions in results, and should be avoided.

(ii) For accurate results, RR and UNI monitors MUST be executed at user priority 3, and preferably with an infinite time slice (both of these things may be set using the CHAP command). SYSPRO automatically has a high priority and an infinite time slice.

(iii) Each monitor takes 1 sample every second, and this determines the number of samples needed to monitor for a desired period.

3.2 Sample rejects

There are two circumstances in which ASHMON monitors reject samples. During sampling two counts of sample rejects are kept by each monitor:

(i) Page fault sample rejects - the number of samples discarded due to a Page Fault occurring in the monitoring process during data collection. SYSPRO is wired into memory, and always returns a 0 Page Fault count.

(ii) Inconsistent disk data sample rejects - the number of samples rejected due to a disk changing status during data collection.

If a significant number (say more than 2%) of samples are rejected, then results could be underestimating the level of disk activity.

3.3 The Overall Control Program

Any monitor may be run directly from its source directory, but the control program gives the simplest interface with the ASHMON package. Its SEG file is MONITOR>ASHMON.SEG. When run, the program presents a menu of all the options within the package.

For a user to run RR or UNI, he must be able to use the CHAP command. If the RR monitor is requested, the user is given the opportunity to CHAP himself and should use a command of the form:

```
CHAP -userno 3 177777
```

which will increase the priority and give an infinite timeslice.

If the UNI monitor is required, then the control program sets up a phantom, CHAPs it to 3, and sends it on its way. The phantom has a como file (ph.YYMMDD.HHMMSS) where any error messages and addresses of unknown semaphores can be found.

N.B. frequent use of the TIDY command is required when using the overall control program as it produces 2 or 3 T$xxxx files every time it is invoked, which are not deleted automatically.

3.4 RR Monitor

Response ratio is the ratio of actual response time (for an interaction, or a set of interactions) to an estimate of the optimal

response time (time with no system competition, and all program pages loaded). Therefore an average RR of 8 indicates that response times are about 8 times as long as they would be under optimal conditions.

The RR monitor is designed to present estimates of response ratio interactively over short periods. When the RR monitor starts it asks the user for the number of samples to be taken in each response ratio report, and the number of reports to be given in the session.

The monitor then begins sampling, and presents reports every time it has taken the required number of samples, until the required number of reports has been presented. (N.B. There is no carry over of data between reports, i.e. data is discarded after each report and recording starts afresh). Reports have the following format:

```
Time : HH:MM:SS
   RR =  r1      Av. number of users active in period =  r2
Disk rejects n1   Page rejects n2
# samples taken n3
```

Where Time = time report generated
RR = Average response ratio in sampling period
Av. number of users active in period = Av number of active terminals and phantoms
Disk and page rejects are as previously described.

3.5 UNI and SYSPRO monitors

UNI and SYSPRO are both monitors which gather data suitable for performance summary preparation and plotting. Each produces data in the same format.

Data files are written into the 'current' UFD when the monitor was started (for UNI), or the 'current' UFD when the data is retrieved (for SYSPRO). Names of data files produced by the monitors have the form :-

SYSPRO - sys.DDMMMYY.HHMMSS
UNI - uni.DDMMMYY.HHMMSS

where DDMMMYY represents the date at the end of the sampling run
HHMMSS represents the time at the end of the sampling run

This system for naming data files may appear rather painful, however it seems the most systematic method to use. After collecting a large number of data files you might appreciate it!

When each monitor starts, the user is prompted for the number of valid samples required for the monitoring session.

4 COMBINING DATA FILES

A program which combines data files produced by the SYSPRO and UNI monitors is provided. Combination of data files may only be done for data files of the same type i.e. data files produced by the same monitor. The combined file is written into the current UFD, and derives its name from the name of the combined file which has the earliest date/time combination. The name of the combined file is the same as the earliest date/time combination in any of the combined files, with a .# extension where # is the lowest available extension for that particular file name in the current UFD in the range 1..9. An errror occurs if all the extensions in ths range are in use.

The information held in the combined file about the files from which it was drawn is :-

earliest time/date of any of the combined files
latest time/date of any of the combined files
number of files combined to make this file

When invoked the combination program queries the user for filenames until a 0 is entered, which indicates the termination of a combination session. The type of the first file sets the type of combined file for the rest of the combination. Error messages are given if :- the file name entered is not found (file names may be full path names), the file type clashes with that of the first file, or if extensions 1..9 are all in use.

5 PRESENTATION OF DATA

Two programs are available under the control program to process data from UNI or SYSPRO.

(i) TORTCOMP produces a performance summary which, among other things, gives a breakdown of response time into its components.

(ii) PLOT, which produces plots of response time components against workload.

5.1 TORTCOMP

An example of the output from TORTCOMP is given in Appendix A. When invoked, TORTCOMP asks the user for the name of the input file. The performance summary produced from the input file is written to the output file :-

rept.DDMMMYY.HHMMSS[.#]

with the date, time and extension (if present) being taken from the input file name.

The report has four sections:

(i) Identification
(ii) Results
(iii) User counts
(iv) Response ratio

5.1.1 Identification

The identification section gives :-
- the system name,
- date/times for the earliest/latest files combined to give the report,
- number of data files combined,
- monitor which collected data,
- total number of samples in all files,
- disk and P/F rejects, as described earlier.

5.1.2 Results

The performance report includes two kinds of result:

- frequency, which is a count of the number of terminals found (across all samples) to be in the specified state.

- percentage, which can be interpretted as the proportion of some total time for which an individual user was in the specified state.

The count of terminals found logged-in (335446 in the example of Appendix A) over all samples (20000 samples in the example) is divided and subdivided as follows:

(1) Response Time - the process corresponding to the terminal is waiting for response from the system.

(1.1) CPU - the process is running on, or queueing for, the CPU. This frequency (43010) is subdivided into running (11785) and queueing (31225), with the queueing frequency further subdivided into the categories of queueing on the ready-list, HIPRIQ, ELIGQ and LOPRIQ.

(1.2) Controller 0, Controller 1 - the process is waiting for a disk transfer. The Controller frequecies are subdivided into running and queueing frequencies. Frequencies are given for controllers 0 and 1 regardless as to whether they exist.

(1.3) QCB waits - process is found waiting for a free disk QCB.

(1.4) Mag tape waits - the process is waiting on the Magnetic tape semaphore.

(1.5) N1 locks - the process is waiting on an N1 lock. (Examples of N1 locks are file, UFD, and transaction locks).

(1.6) Network waits - the process is waiting for response from a network.

(1.7) Others - process is found waiting for event not covered in any other category.

The percentages give the proportions of response time attributable to each of the above causes.

(2) Input - the process is waiting on the terminal input semaphore. The time in this state can be loosely interpretted as "think and input" time.

(3) Output - the process is waiting on a timer. There are a number of different reasons for which terminal processes wait on timers, the usual one being that the terminal process is waiting for the terminal buffer to empty.

(4) Others - the process is waiting on a user or numbered semaphore.

Following the response time section is a section of more detailed information about disk IO. A sub section is allocated to each controller. Information is printed for every disk for which activity was detected during the sampling period. Running and queueing frequencies are printed for each drive, as well as the percentage of time that these frequencies make up of total controller time. Running and queueing frequencies are also given for each logical area on every active drive, as well as the percentage that these frequencies are of total time on that drive.

5.1.3 User counts

On each sample several different counts of users are made, and the average of each count is given. The counts are :-

(1) Logged-in users - this count is of the average number of logged-in users, and is subdivided into logged-in terminals, and logged-in phantoms.

(2) Active users - on each sample, a count is made of the number of active user processes. (An active process is one waiting for system response). The average active user count is also subdivided into terminal and phantom components.

5.1.4 Response ratio

Finally an estimate of average response ratio is given. Response ratio for a set of interactions is defined as the ratio of the sum of their actual reponse times to the sum of the shortest possible times which the interactions could have taken. It is estimated here (and in RR) by:

$$\frac{\text{Total response time}}{\text{CPU running + file IO (non-paging IO) running}}$$

5.2 PLOT

PLOT provides a flexible tool to produce graphical output, in either text file or NCAR meta file form. (If NCAR is not available, the code provided may be useful to show how to interface the ASHMON plot routines with a different plotting package).

Plots will be put into frames in files with names of the form ncar.DDMMMYY.HHMMSS.# for NCAR meta files, and line.DDMMMYY.HHMMSS.# for line printer text files. The date/time combination used in the naming of the plot files is the earliest date/time combination specified in the input file. # is the lowest extension in the range 1..9, where no other files in the current UFD have the same name for either the NCAR or LINE filenames. If both NCAR meta files and line printer text files are produced by the same plotting session they are given the same extension number #.

PLOT allows:

(i) On the X-axis, either of two measures of workload :-

(a) Logged-in terminals, or

(b) Active users (average number of users, terminal and phantom, active over the last 5 samples)

The UNI and SYSPRO monitors take these counts on each sample, and the response data collected on each sample is put into two arrays using the above counts to index the two arrays.

(ii) On the Y-axis, quantities typically relative to the times which terminals spend in different states. These can be displayed as

(a) The average number of terminals per sample found in that state, or

(b) Percentage of response time spent in that state, or

(c) Percentage of total time spent in that state.

5.2.1 Using PLOT

The menu structure of the plot package user interface allows the user to plot quantitites to the same level of detail as there is in the response time breakdown report. An example of a plot session, showing some of the different menus, is given in Appendix B.

When PLOT starts executing the user is prompted for the name of a SYSPRO or UNI data file (name may be a full path name), and prompting continues until a valid path name is given. Plot files are written into the current UFD.

The user is then asked to specify the plot file type, the workload measurement for the X-axis, and the units to be used on the Y-axis. The main (root) menu is then entered. From this menu the user can :-

(i) Plot components of total time : response time (menu), think time, waits for terminal buffers to empty, waits on user and numbered semaphores, and total response time.

(ii) Plot the numbers of logged-in or active phantoms,

(iii) Plot response ratio, *

(iv) Enter utility menu,

(v) Plot the frequency or cumulative frequency distributions of the number of samples, for the present user measure.

(vi) Terminate the current frame, or exit plotting session.

* response ratio may only be plotted when Y-axis units are av/sample.

N.B. when the first plot is made to a frame, and the Y-axis units are av/sample, the user is queried for the upper limit to give to the Y-axis. If any points subsequently plotted exceed this limit a warning is given and offending points are given a Y value equal to the current upper limit.

Plots of cumulative frequencies give a cumulative frequency for the distribution of the number of samples falling into each interval for the current user measure. Plots of data density show the percentage of the total sample which falls into each interval. These plots should give a good indication of where the data is densest, and the results most reliable. When either of these plots are done the old plot frame is terminated, the plot put into the new frame, and that frame is then closed.

The utility menu gives access to the menus which were gone through initially, and enables the changing of :-

- the current plot file,
- the current user measure,
- the current Y-axis measure.

If any of these changes are made then the current frame is terminated.

From the response time menu you may :-
- enter the CPU menu,
- enter the disk IO menu,
- plot any of the minor response time components,
- plot total response time.

The CPU menu allows plotting of running, total queueing or total CPU time, as well as any of the components of CPU queueing time (time on ready-list, HIPRIQ, ELIGQ or LOPRIQ).

The disk menu allows you to
- plot running, queueing or total disk IO time,
- enter paging or non-paging disk IO menus,
- enter the menu for either disk controller.

The paging and non-paging menus allow plots of running, queueing or total time spent on paging or non-paging disk IO requests.

Controller menus allow plotting of running, queueing and total time spent on that controller, as well as giving entry to menus for each drive on that controller.

Drive menus allow plotting of running, queueing and total time spent on that drive, as well as giving entry to menus for each logical area on that drive.

Logical area menus allow plotting of running, queueing and total time spent on that logical area.

N.B. all plots are done in the context of the current X and Y-axis measures, into the current plot frame.

5.3.2 Plot frame format

Plot frames contain the same identification as a response time breakdown report. They also contain labelled axes with scales, the plotted points themselves, and (up to 6) descriptions of plotted curves. The first curve plotted in any frame will have 'A' as its plot character, the second 'B', and so on. Any number of curves may be plotted in a frame, but a maximum of 6 pieces of text explaining a plot charcter may appear on any frame. Any points whose Y values exceed the upper limit of the Y-axis for the current frame have their Y values set to the upper limit.

The current frame is terminated by user request, or when the current plot file, Y-axis measure or X-axis measure is changed, or by plotting a cumulative frequency or data density curve.

6 GETTING STARTED

Initially configure your system by running MONITOR.BUILD.CPL (Any subsequent reconfigurations may be done via the overall control program). It is probably best to start out using the UNI monitor and gaining experience in using the utlities, especially the plotting package.

When serious experimentation is done a sub-UFD dedicated to data files would be useful. The monitoring should be started at the same time each day, and the same number of samples taken (if monitoring is done over a number of days).

If you wish to be more adventurous then you can include SYSPRO in your operating system, and modify the plotting package to use any special plotting packages you have on your system (using MONITOR>PLOT>NCAR.PASCAL as a blueprint for a file of subroutines to interface with your plot package, and making the necessary changes to MONITOR>PLOT>PLOT.PASCAL).

Good luck! Any problems contact Paul Ashton, Comp. Sci. Dept, University of Canterbury, Private Bag, CHCH, New Zealand.

Appendix A

In this Appendix is an example of a Performance Summary. The sampling done for this particular report was done at the University of Canterbury during a period of light-medium usage.

Performance summary

1. Identification

System sampled :- University of Canterbury P750
Date of first sampling file
   Date of sample :- 13 AUG 84   Time sampling finished :- 15:26:41
Date of final sampling file
   Date of sample :- 13 AUG 84   Time sampling finished :- 15:26:41
Number of files combined :- 1
Monitor used :- UNI
Samples taken :- 20000
Samples discarded through inconsistent data :- 79
Samples discarded through P/F during sample :- 38

2. Performance results

Response time

CPU	43010	85.9	Running	11785	
			Queueing	31225	->Ready-list 16092
					HIPRIQ 2528
					ELIGQ 9548
					LOPRIQ 3057
Controller 0	4179	8.3	Running	2719	
			Queueing	1460	
Controller 1	1725	3.4	Running	1316	
			Queueing	409	
Waits QCB's	0	0.0			
Other peripherals:					
Mag tape	126	0.3			
Nl locks	1040	2.1			
Network waits	0	0.0			
Others	11	0.0			
		100.0			
Total response time	50091	14.9			

Input

Think and input time	275296	82.1

```
 Output
 ------
   Waits for term buffer 10059     3.0

 Others
 ------
   Waits on user sems         0     0.0
                          -----   -----
Total time                335446   100.0

 Further analysis of disk traffic

 Controller 0

  Disk 0   Running  :-  2719    65.1
           Queueing :-  1460    34.9
   CANT00        Running  :-   824    19.7
                 Queueing :-   565    13.5
   PAGDEV        Running  :-  1053    25.2
                 Queueing :-   515    12.3
   CANT01        Running  :-   842    20.1
                 Queueing :-   380     9.1

 Controller 1

  Disk 0   Running  :-  1299    75.3
           Queueing :-   408    23.7
   CANT02        Running  :-   292    17.1
                 Queueing :-   110     6.4
   ALTDEV        Running  :-   702    41.1
                 Queueing :-   227    13.3
   CANT03        Running  :-   305    17.9
                 Queueing :-    71     4.2

  Disk 1   Running  :-    17     1.0
           Queueing :-     1     0.1
   CANT04        Running  :-    17    94.4
                 Queueing :-     1     5.6

Average number of logged in users :-  34.8
   Average number of terminals logged in :-      16.8
   Average number of non-terminals logged in :-  18.0
Average number of active users :-   3.9
   Average number of active terminals :-       2.5
   Average number of active non-terminals :-   1.4

Mean Response ratio =  3.56
```

Appendix B

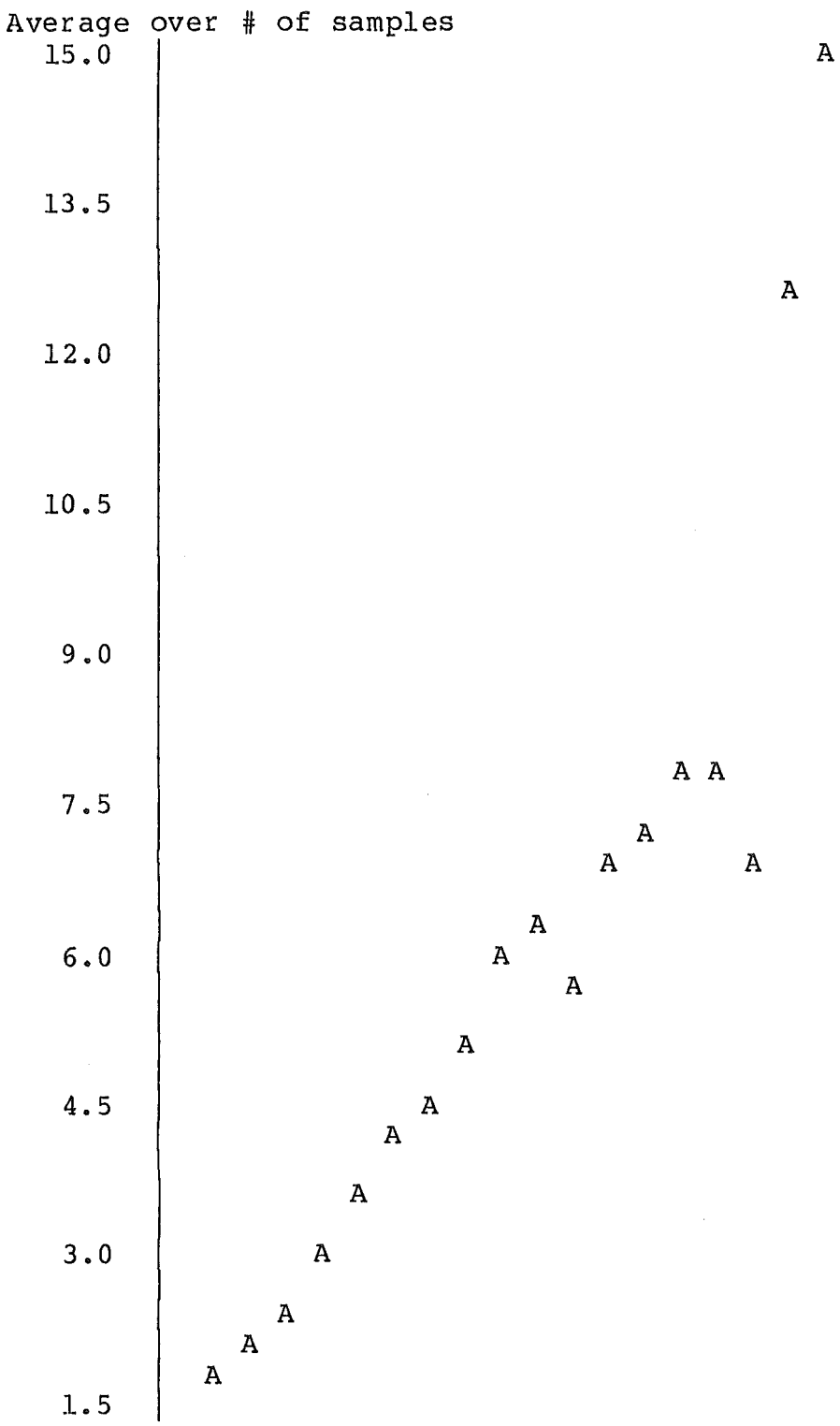
In Appendix B a simple example of using the plot package, via the overall control program, is given. This plot was taken from the same data file as the RT breakdown report given in Appendix A.

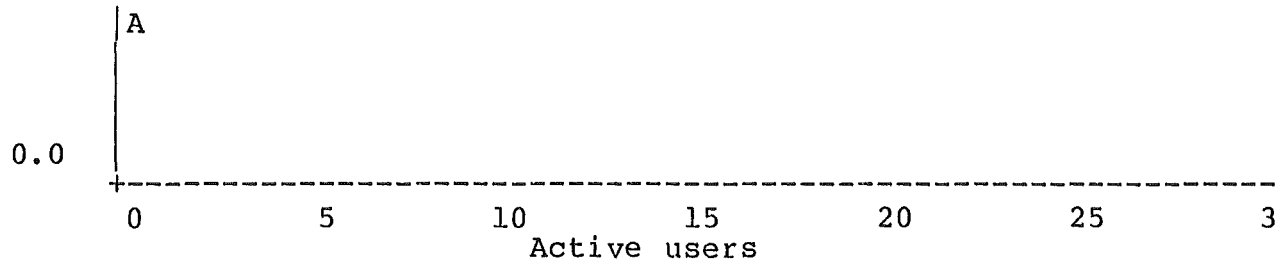
```
OK, ashmon
[ASHMON Ver 1.0]
Welcome to the ASHMON Prime monitoring package.
And what do you want to do today?
   1. Run UNI monitor.
   2. Start a SYSPRO monitoring run.
   3. Retrieve data from a SYSPRO run.
   4. Run RR monitor.
   5. Enter plotting package.
   6. Convert data to RT breakdown.
   7. Combine data files.
   8. Configure ASHMON system.
   9. Quit.
Enter number: 5
OK, CPL T$0000

Enter input file name: uni.13aug84.152641
Line printer plots written to :- line.13AUG84.152641.1
NCAR plots plotted to :- ncar.13AUG84.152641.1

Plotting choices:
Plot file selection.
   1. Produce NCAR plots.
   2. Produce line printer plots.
Enter number: 2

Plotting choices:
User count selection menu.
   1. Active users.
   2. Logged in terminals.
Enter number: 1

Plotting choices:
Y-axis unit measurement menu.
   1. Average users.
   2. Percent of response time.
   3. Percent of total time.
Enter number: 1

Plotting choices:
Menu 1 :- Master menu
   1. Response time
   2. Input time
   3. Output time
   4. Numbered and user semaphores
   5. Total time
```

```
   6. Logged in phantoms
   7. Active phantoms
   8. Response ratio
   9. Use utility menu
   10. Produce cumulative frequency
   11. Show distribution of samples
   12. Terminate current frame and begin another
   13. Quit
Enter number: 8

Enter upper bound for Y-axis: 15

Plotting choices:
Menu 1 :- Master menu
   1. Response time
   2. Input time
   3. Output time
   4. Numbered and user semaphores
   5. Total time
   6. Logged in phantoms
   7. Active phantoms
   8. Response ratio
   9. Use utility menu
   10. Produce cumulative frequency
   11. Show distribution of samples
   12. Terminate current frame and begin another
   13. Quit
Enter number: 13
OK, COMI -E
OK, como -e
```

Resulting plot

Sampled :- University of Canterbury P750 First date 13 AUG 84
at 15:26:41 Final date 13 AUG 84 at 15:26:41 Samples 20000
Disk rejects 79 Page rejetcs 38 Files combined 1 Monitor :- UNI





A Response ratio

Appendix C

This appendix is a brief guide to anyone wishing to examine the source code of ASHMON. The different programs in the package generally occupy their own directories.

The MONITOR directory itself contains the programs :-

ASHMON - the overall control program

NF and WT - programs to provide co-ordination between the terminal and monitoring phantoms.

There are also some generally used Pascal programs in MONITOR :-

GETNUM - contains routines to read positive integers and character strings from the terminal.

LIB - contains routines to fill in the date and time in a data file name, and to write a line of data to SYSPRO and UNI data files.

CALL - a dummy Pascal program to call monitors.

MONITOR also has the sub-UFD's :-

COMBINE - combination program,
CONFIG - the configuration program,
DOC - user documentation,
INSERT - const, type and var insert files for PLOT and CONFIG,
- insert files with disk information for Pascal programs,
- template files for CONFIG,
- general PMA and Pascal insert files,
PLOT - plot package,
RR - RR monitor,
SYSPRO - programs to implement, and interface with, SYSPRO,
TORT - report generation program,
UNI - UNI monitor.

In each UFD there are also CPL programs for loading (names end in LOAD) any programs in that UFD, and CPL programs for compiling and then loading (names end in COMP) any programs in that UFD.